Michael Solomon Desta

A Local Pedestrian Mobility Model for Urban Content Sharing

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo January, 2013

Thesis supervisor:

Professor Jörg Ott

Thesis advisor:

Professor Jörg Ott

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Michael Solomon Desta

Title: A Local Pedestrian Mobility Model
for Urban Content Sharing

Date: January, 2013 Language: English Number of pages:7+84

Department of Communications and Networking

Professorship: Networking Technology Code: S-38

Supervisor: Professor Jörg Ott

Advisor: Professor Jörg Ott

In opportunistic content sharing schemes such as the *Floating Content Concept*, a piece of content deemed to be relevant in certain geographical region (anchor zone) is made available for sharing without the support of any kind of infrastructure. Using the principles of opportunistic networking, content is stored and replicated among mobile nodes, ultimately dwelling in the anchor zone in a scrounging manner. As a best effort scheme, availability of content depends on the availability of information carrying nodes and their movement behaviour within the anchor zone.

In line with capturing important pedestrian mobility patterns, the research community have proposed and studied a wide range of mobility models for ad-hoc networks. For opportunistic content sharing, *microscopic level* mobility models that capture individual movement patterns are important. At the *microscopic level*, we have an "open" simulation world in which nodes enter and leave rather than a closed system in which nodes reside permanently. Moreover, these "open worlds" such as city squares often have small number of concurrent nodes which makes steady state analysis insufficient.

In this thesis work, we investigate the operation of a content sharing application, Floating Content, under such microscopic mobility conditions and characterize its behavior for city squares. For its validation, we introduce the *Square Mobility Model*, a pedestrian mobility model for content sharing in open city squares. We use stochastic modeling techniques to derive a mathematical expressions for capturing important elements of pedestrian movement in squares.

Keywords: Delay Tolerant Networking, DTN, Opportunistic Networking, Best effort, Content Sharing, Mobility Model

Preface

First and foremost, Praise to The Lord, the almighty GOD for "The fear of the LORD is the beginning of Knowledge".

This Master's Thesis has been done to the Department of Communications and Networking (Comnet) at the School of Electrical Engineering, Aalto University. I would like to express my gratitude for the people who has made this thesis work possible.

My deepest gratitude goes to my supervisor, Professor Jörg Ott, for giving me the opportunity to work under his supervision and for all his inspirations, guidance and encouragement. He has been very friendly, understanding and supportive in all dimension.

I want to express my gratitude to Dr. Esa Hyttiä for all the invaluable discussions and support he gave me during this thesis work. My gratitude also extends to Dr. Mikko Pitkänen and Ari Keränen for their help in the ONE.

My special thanks goes to my wife Fev. Without your patience and support this work would have been next to impossible to complete. And Adiye, welcome and we love you.

Finally, I want to thank all my family for being there and I dedicate this to you all!

January, 2013 in Espoo, Finland

Michael Solomon Desta

Contents

List of Abbreviations

2D	Two Dimension
ABS	Agent-Based Simulation
AP	Access Point
ARPA	Advanced Research Project Agency
CRAWDAD	Community Resource for Archiving Wireless Data At Dartmouth
DARPA	Defense Advanced Research Project Agency
DES	Descrete Event Simulator
DoD	Department of Defense
DTN	Delay/Disruption Tolerant Networking
GUI	Graphical User Interface
LBS	Location Based Service
MBM	Map Based Mobility
MN	Mobile Node
OM	Obstacle Mobility
ONE	Opportunistic Network Environment
PDA	Personal Digital Assistant
RPGM	Reference Point Group Mobility
RTT	Round Trip Time
RWP	Random Way-Point
SPMBM	Shortest Path MBM
TTL	Time To Leave
WLAN	Wireless Local Area Network
WORN	Write-Once, Read-Never

1 Introduction

Over the past decade we have witnessed the explosive growth in the use of PCs, Internet and personal communication devices in our day-to-day life. Advancements in the software and communications technology have enabled revolutionary applications such as Face-book, Twitter, YouTube and other social Medias which have proven to be widely popular and essential in our day to day life . Similarly, manufacturing in Integrated Circuits and electronics technology has become even more sophisticated by the day that it is now possible to cheaply and easily integrate millions of transistors and circuit components into a single hand-held device such as smart phones, digital cameras, and PDAs. These advancements have resulted in portable devices with higher computing and storage capacities, as well as various communication capabilities. The famous Moore's Law in his 1965 publication "*Cramming more components onto Integrated Circuits*" stated that [1]:

"*The complexity for minimum component costs has increased at a rate of roughly a factor of two per year... Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000. I believe that such a large circuit can be built on a single wafer.*"

In fact, we can argue that Moore's law holds even today.

With the introduction of the Internet, web 2.0 and the power of integrated circuits and Communications technology, the notion of physical distance being a "barrier" for our everyday life is diminishing. Moreover, the increase in the number of wireless communication devices and improvements in access schemes has made content sharing an easy and comfortable practice. Voice and video calling between far ends of the world, sharing pictures with our friends and family over the Internet and social Medias, following news and latest developments globally are now among the common place practices of modern society.

While such services are useful in bridging physical distance, on the other hand, we have Location Based Services (LBS) in which the geographical location of a user is used as an input for using certain features and service of LBS applications. These services, such as local information searches and 'check-in' apps are now becoming popular and quite useful. Currently, all these features and services are mainly enabled by the use of (existing) communication infrastructures such as our mobile phone network operator and Internet Service provider. But, relying on infrastructure based services is not always a good idea. Building communication infrastructure for communication network coverage is both time consuming and expensive investment. Companies investing in telecommunication and Internet infrastructure are always selective as where to build a certain kind of infrastructure and the business case involved with it. For example, sparsely populated rural areas have less network coverage and communication infrastructures as compared to densely populated urban areas. Regions with extreme terrestrial environments, harsh weather conditions or planned networks in space have also proven to be difficult to build infrastructures.

Proper functioning of infrastructure networks could also be challenged by natural phenomena such as environmental disasters, flooding, earth quakes & tsunami and so on. For example, in a city stricken with an earth-quake, rescue and disaster recovery attempts will be hampered by the loss of mobile network coverage or power outages.

Limitations of infrastructure based communication networks becomes pronounced even more when using LBS, for example for sharing a message or content which is only locally significant. This doesn't sound appealing whereby current communication technologies allow nearby user devices to communicate and form a network of their own without relying on any kind of existing infrastructure. These kinds of networks are commonly known as Ad-Hoc Networks in which the communicating devices can act both as a router and end-user device to form an infrastructure-less network.

In more extreme environments where nodes are sparsely distributed and the resulting lack of end-to-end connectivity requires nodes to carry packets for a certain time before the packets can be forwarded to other nodes closer to the destination form *Delay-Tolerant Networks* (DTN) [2] [3].

1.1 Delay Tolerant Networking

If not all, most of today's applications and services in the Internet are built on top of pre-existing communications network infrastructures. Moreover, the Internet is built on the notion of global reachability, where end-to-end connectivity between peers is assumed to exist. But, in extreme and performance-challenged environments such as space communication or sparsely populated regions, continuous end to end connectivity is not always guaranteed. This is problematic as protocols designed for otherwise convenient networks may not perform well. Delay/Disruption Tolerant Networking (DTN) is a networking effort used to address such problems by providing inter-operable communications among highly heterogeneous networks that lack end to end connectivity.

DTN, now established as one of the mainstream research areas in communications and networking, takes a different approach to (inter)networking and allows working in stressed as well as in highly heterogeneous environments. Historically, research in DTN type networks dates back to 1958 when the United States Department of Defense (DoD) responded to the launch of Sputnik by the Soviet Union and created the Advanced Research Projects Agency (ARPA) and later on DARPA (Defense Advanced Research Projects Agency) [4]. But at that time, computers were only making the transition from vacuum tube based to transistor based architecture and the research was mainly limited to short range communications for human space flights, and communications between earth and orbiting satellites. However, by the 1990s, computers evolved to serve as general purpose machines with Integrated Circuit based architectures, which also resulted in miniaturized and mobile computers. It was then that the fields of *"mobile ad-hoc routing"* and *"vehicular ad-hoc networking"* starts gaining momentum adding the flavors *"Delay"* and *"Disruption"* into the research community.

Over the years, networking and communications technology grew tremendously

in line with improved access schemes, better routing and switching capabilities and high speed bandwidth availability. At the same time, applications in such complex and dynamic networking environment need to operate in various obstacles that stem from high delay in inter-planetary networks or variable bandwidth across the Internet, power outages and lack of sufficient communication infrastructure in rural or sometimes extreme environments. Therefore, when access to traditional network infrastructure is obscured, the network is deemed to be challenged network. Established to address these challenges, the potential usage of DTNs and Ad-Hoc networks for various applications such as in the military, environmentally hazardous areas, and regions stricken with natural disaster has increased recently, which is also apparent by the growing interest of research in such networks.

DTNs can also be considered as a good example for opportunistic networks, as these networks attempt to overcome the shortages of nodes (mobile) in messaging and communication based on spontaneous/opportunistic connectivity between users with wireless devices. In broader sense, an opportunistic network is a network consisting of nodes which are equipped with short-range radio transmission functionality and wirelessly connected to each other. An opportunistic network node can be either mobile or fixed.

1.2 Floating Content

Over the past 5 -10 years, opportunistic ad hoc content sharing has attracted a considerable amount of attention within the academia and industry as an alternative to infrastructure supported LBS and content sharing scheme. Opportunistic ad hoc content sharing offers one to many attractive properties as compared to the infrastructure based variants.

Apart from the limitations and vulnerability of using infrastructure based services mentioned above (chapter 1), few more restricting factors for the use of such infrastructures for location based services are discussed in [5–7]. Among these factors, user location privacy is one, where a user does not need to report its location for service provider, which is otherwise true. Other benefits of infrastructure less opportunistic networks include:

- Connectivity: In order to use currently available location based services, it is obviously a prerequisite to connect to some kind of infrastructure in that location. From the point of view of a traveling person (e.g. tourists), such connectivity is faced with limiting factors such as high roaming costs, unavailability of data services or even to the extent of no network coverage at all.

- Location privacy: Using infrastructure based services, to get the right context of information for a certain location typically involves communicating one's location to some sort of remote server. This may not always be desirable, which also raises the question of location privacy.

- Content privacy: Often, shared information in the Internet is stored at some "central location". Although centralized content management offer its own advantages, content in such locations are exposed to different type of attacks and censorship.

- Geographic validity: Not every piece of content is relevant in every corner of the globe. Certain content relevant to a specific location could rather be stored locally and accessed in the same way.

- Temporal validity: Data centers and storage facilities are populated with a huge chunk of WORN (Write-Once, Read-Never) content that was once written in the past but never got the chance to be read. One means to reduce WORN contents is to tag contents with an expiry time, so that they can be removed when they are no longer necessary.

In response to the above mentioned limitations and opportunities, the research community has proposed and studied many different variants of opportunistic content sharing schemes, e.g., [5–11]. One such variant is the floating Content, which is a best effort content sharing scheme. The idea is that, a geographically significant content item created by a local user is made available in that location (anchor zone) without the support of any kind of infrastructure. This is made possible by the use of opportunistic content sharing mechanism, which solely depends on the existence of interested mobile users (nodes) within the anchor zone [7].

In subsequent sub-sections, we will discuss the basic concept of floating content and its system operation. We refer interested readers to [5–7] for a more detailed description of the floating content concept.

1.2.1 Basic Concept

Let us consider a given geographical region or anchor zone as in figure (1) where mobile users (nodes) enter the region, spend some random time there and leave. A user with a piece of content (seed node) wants to share his/her content with visitors of that specific region, but without the use of any kind of infrastructure. This seed node will "post" its content, and interested nodes visiting the anchor zone will get a copy of the content when they come in contact (within the communication range) with the node having the content. Mobile phones or other similar devices with enough storage capacity and relevant wireless interfaces (e.g. WLAN or Bluetooth) can be used to "post" and share content. As a result, a given content will replicate and pass onto new visitors of the anchor zone in an epidemic fashion, even when the original seeder node has left the area.

As noted above, interested nodes keep copies of information items floating around the anchor zone by replicating the items when they meet. Hence, the floating content does not rely on any infrastructure, rather on the availability of enough nodes. As we allow all nodes (including the original seeder node) to leave the anchor zone randomly and delete the content when they exit, the availability of content will be probabilistic, and the system can be regarded as a best effort system. Eventually, when there are

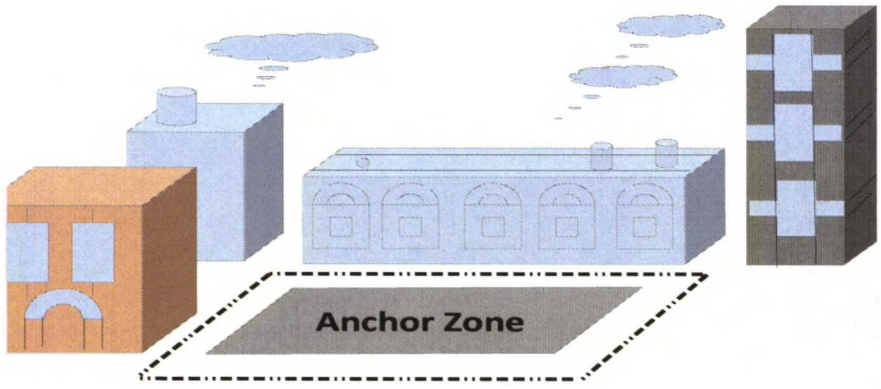


Figure 1: Floating Content in a typical City-Square.

no nodes in the region with the corresponding content, the information will be lost irreversibly. When this happens, we say the content has "sunk". Additionally, content items can tagged with a time to live (TTL) and discarded upon expiry.

1.2.2 System Operation

Floating content operate when nodes with presumably relevant content in a given geographical area (e.g. image or local advertisement), opportunistically share content with nearby nodes. Initially, a node (seed node) having a piece of content will generate information item I and "post" it for sharing in the anchor zone. Together with the actual content, item I contains anchor zone information (replication and deletion region), content life time, and meta-data that make it easier for other nodes to filter and search for content they are interested in. Since the system does not use any kind of existing infrastructure, the original node must physically be in the anchor zone when posting its content.

Content will start to replicate within the anchor zone when other nodes come in contact with the seed node. In general, when two nodes A and B meet in the anchor zone of an item I, and A has I while B does not, then A replicates it to B. Nodes leaving the anchor zone are free to delete their copy of the item. In the floating content concept, it is generally assumed that content is unconditionally deleted outside the anchor zone, whereas the replication occurs only inside the anchor zone.

1.2.3 Criticality and Finite Systems

Content replication within the anchor zone is directly dependent on the contact rate of nodes which itself depends on the transmission range and the locations of the nodes, dictated by a scenario specific random mobility pattern. Due to the stochastic nature of node contacts, it is not always clear when floating content can be supported or what the necessary conditions are. But assuming there is sufficient number of nodes in the anchor zone, availability of content can be guaranteed with certain finite probability. Information availability in floating content has been studied such

as in [12] and [13]. In these studies, the so-called *criticality condition* which is the necessary condition to float a piece of content in a given geographical area has been established.

The criticality condition defines, for large systems (at fluid limit), the necessary conditions (in terms of node density, transmission range, anchor zone, etc) under which a population of mobile nodes can support the floating content. In a non-spatial black-box model, this condition takes the form [5]:

$$\frac{2R}{\mu} > 1 \quad \Leftrightarrow \quad \frac{N\nu}{\mu} > 1, \tag{1}$$

where R is the total contact rate of nodes in the anchor zone, μ the node arrival/departure rate, N the (mean) number of nodes in the anchor zone, and ν the rate at which an arbitrary pair of nodes meets each other in the anchor zone. More detailed descriptions taking into account also the spatial dimension can be found from [5].

In reality, there are usually a small number of nodes in a typical small anchor zone (i.e., the content is highly localized), and stochastic fluctuations in the node population have to be taken into account. In fact, a floating content in the real world is constantly in a transient towards extinction, which would happen latest, e.g., during a night time. However, each content item has also a user defined finite lifetime and, in best-effort fashion, it is adequate that, with a sufficiently high probability, the published content remains available until it expires.

Essentially, the content survival in a large system relies on averages as stochastic fluctuations become negligible at macroscopic scale. However, this does not apply to a single node and indeed at the criticality threshold only a small fraction of the nodes have the content item. In particular, when a new content item is created, it is carried (by definition) by a single node and it is of uttermost importance that the seeder manages to pass the content item to several nodes[1] [ref]. Otherwise, the content is likely to disappear when the seeder leaves the anchor zone. We refer to this phase as *bootstrapping* and its investigation is also one of the key contributions of this thesis work.

Moreover, it should be noted that criticality condition tells us only if floating content is sustainable, i.e., if first successfully distributed across the anchor zone, then the content will exist for long periods of time (infinitely at the fluid limit) whenever the criticality condition is met. However, in practice it is rarely sufficient that the content is carried by a small fraction of nodes somewhere in the anchor zone. Instead, we rather require that the content must be "easily" available, which is achieved when a certain strictly positive fraction of the nodes carry it.

1.3 Disposition

The structure of the remaining thesis is as follows. In chapter 2, we will present and discuss a wide range of mobility models for ad-hoc networks. Then in chapter

[1]The system cannot enforce this, thus a sufficient number of nodes must reside in the anchor zone to ensure content availability.

3, we will define the research question of this thesis project and methodologies used to solve these problem.

In chapter 4, we will introduce the *Square Mobility Model*, a local pedestrian mobility model for urban content sharing. We will derive the mathematical expressions for modeling pedestrian movement in city squares, their contact rate and stationary spatial node density. Mathematical analysis of *floating content* with small number of nodes and when the system is initialized for the first time is presented in chapter 5.

Chapter 6 describes the Opportunistic Network Environment (ONE), the simulator we used to implement the *square mobility model* and then present the implementation of our model in the simulator. In chapter 7, simulation result of *floating content* in *city squares* will be discuss by comparing them with the theoretical analysis from previouse chapters. Finally, chapter 8 will conclude the thesis work by discussing the main findings and limitations of this work, and some suggestions for future work.

2 Mobility

In order to maintain connectivity and relay messages, opportunistic networks rely on mobile devices that can be found nearby. Typically, such mobile devices are carried by humans [14]. As a result, the design and performance analysis of opportunistic networks is highly dependent on the use of a carefully crafted human mobility model. A number of studies such as [14] and [15] have shown that user mobility patterns affect the performance of DTN protocols and applications to great extent. Thus, in order to properly evaluate a given system which is dependent on the mobility behavior of its users, it is vital to accurately capture the real and important characteristics of such mobile users with the help of a mobility model that can mimic the patterns. A good understanding and modeling of user mobility behavior is also important because these models are the raw materials for simulation tests which are usually the next steps in the course of studying opportunistic networks. The design and analysis of the opportunistic content sharing scheme introduced in section 1.2, Floating Content, is no different as it is inherently dependent on the existence of mobile nodes and their movement patterns which needs to be modeled accurately if the system is to support content sharing without the use of any kind of existing infrastructure.

In this chapter, we will introduce different mobility models proposed for performance evaluation of networks protocols and services. We will try to cover a wide variety of models ranging from legacy mobility models such as the *Random-Way Point* upto the more recent models like the *Working Day Movement* models[2]. Before we go any further and discuss these models, let us define some of the metrics and properties used to characterize mobility models.

2.1 Characterizing Mobility Models for Ad-hoc Networks

Until recently, research efforts into understanding user mobility concentrated at a *macroscopic level*, such as mobile user cell changing rate or call blockage probability. Such models are often used to describe the movement patterns of nodes at large, often for the purpose of cellular network planning purposes or to assess feasibility of different communication on different locations. Therefore, *macroscopic mobility models* lack granularity at the level of individual nodes.

With the growing popularity of ad-hoc and DTN networks, the focus is shifting towards understanding the microscopic movement behavior of nodes, like the relative velocity and location of nodes with one another. The shift was important because such microscopic factors determine when links are formed or broken, and for how long. In light of these changing needs, quite many mobility models have been put forward and used for the evaluation of protocols and services for ad-hoc networks. All mobility models have their own history of evolution and intended area of application. As such, a number of studies such as [14] and [15] have shown

[2]**Disclaimer**: The models we discuss in this thesis work are only meant to give the reader some intuition about the availability of a wide range of mobility models, and it is by no mean a complete list of all mobility models

that using the wrong mobility model for performance evaluation of ad-hoc protocols might lead to a disastrous result. However, careful examination of a model and its properties could reveal where it is appropriate to use it and where not leading to a more tangible result.

2.1.1 Mobility Metrics

One means to understand the properties and applicable area of a mobility model is the set of metric/variables it uses to control movement behavior of nodes. This is because researchers often conduct sensitivity analysis to determine the performance of a protocol or network service under different conditions, for example robustness of a system under extreme setting of the variables or to find out the optimum performance point. Below are some of the most common mobility metrics used to characterize mobility.

Inter-contact Times - Also known as *inter-meeting time*, is a variable that holds the time period for which a pair of node are not in contact. This metric is used to quantify the formation or breakage of opportunistic links that can be used to send packets between a pair of nodes. Often, the distribution of the *inter-contact times* is used to understand these opportunities.

Contact Times - Contact times, or sometimes referred to as *contact duration* is the duration of a contact that last between nodes. It is an important quantity because in the absence of interference from other nodes, for example in sparsely populated networks, the contact duration is often the limiting factor as how much data can be transferred.

Contact Rate - is defined as the number of contacts a node makes per unit time. This metric, together with other relevant ones, could be in a decision to select the node having the most contact to forward a message to its destination.

2.1.2 Mobility Area

Yet another important factor to characterize mobility models is the (simulation) area where nodes move. Different mobility models assume different mobility area based on shape, size and dimension (often 2D). Some also characterize the behavior of nodes in the extreme limit of the area, for example around the edge of the mobility area. One example is the *boundless simulation area* mobility model [16]. One unique feature of this model is how it assumes the simulation boundary. While other mobility models assume a 2D simulation area with edges that cannot be crossed, the boundless simulation model assumes a torus-shaped form of the 2D area so that nodes don't have to stop when they reach the boundary. A node that reaches a boundary will leave the region and come in again in the opposite edge of the region. For example, a node that reaches the top-middle edge of a rectangular simulation area will not bounce back or just disappear, but come into the simulation through the bottom-middle edge of the region. Another is *City Section Mobility Models* such as the *Manhattan model* [15] which also takes a different approach to the simulation area of mobility models by assuming cross-section of a road within a city.

2.1.3 Classification of Mobility Models

With the ever increasing complexity and variety of mobility models for ad-hoc networks, it has become typical of researchers to first survey and study the availability of existing models for their intended networks. A survey and classification of existing mobility models for ad-hoc networks is nicely presented, for example in [17] and [18]. As presented in [18], one simple way to study mobility models is to first classify them as entity or group mobility models. In this type of classification, models that fall into the entity category consider a single mobile entity and assume that the movement pattern of one mobile node is independent of other nodes in the vicinity. Some of the most commonly used entity mobility models for performance evaluation of ad-hoc networks include *Random based Mobility models* such as *Random Walk* and *Random Way Point* mobility models, *Boundless simulation Area Mobility* Models, the *Gauss-Markov mobility* model and *Pathway mobility* models.

On the other hand, there are certain mobility models that assume nodes move as a group or make movement decisions such as direction or speed together. These mobility models are collectively known as *group mobility models*. Unlike entity mobility models, a single mobile node's movement depends on how other nearby nodes are moving, for example going to a certain target location as a group. Figure (2) summarizes some of the most commonly found synthetic mobility models and their classification as group or entity models.

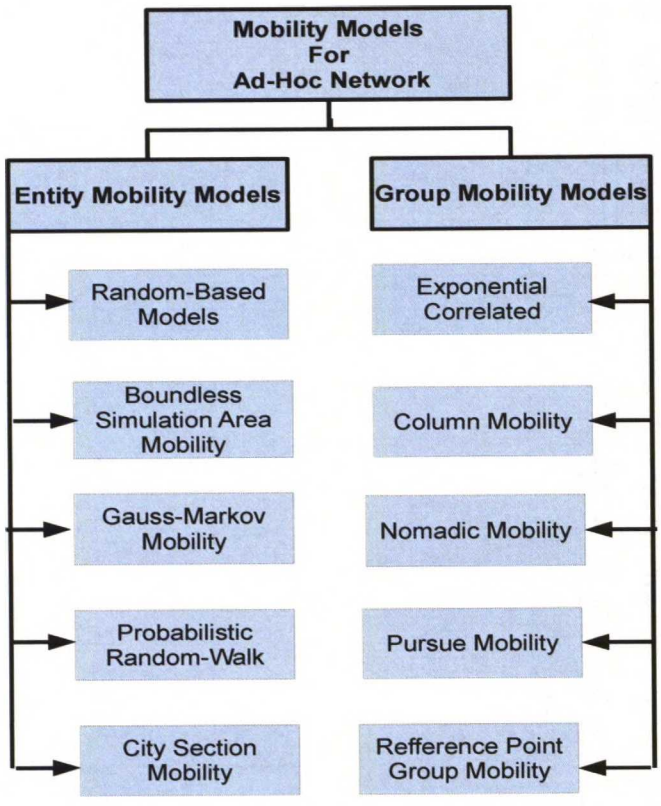


Figure 2: Classification of synthetic mobility models as group and entity model.

A different approach to study and classify ad-hoc mobility models is also presented in [17]. In this paper, a wide variety of mobility models are categorized based on specific characteristics they share such as *temporal* or *spatial dependencies, geographical restrictions* and *randomness* of movement patterns. Models with *Temporal dependency* have memory, meaning previous movement history may influence current movement pattern of a mobile node. *Gauss-Markov* and the *Smooth Random* mobility models are typical examples of mobility models with temporal dependency. In some other mobility models, nodes tend to follow a certain pattern of movement as a group, showing the property of *spatial dependency.* Mobility models that consider different forms of geographical restrictions and obstacles in the simulation area are classified as models with *geographic dependency.* Yet, another class of mobility models is the random models that assume complete randomness of single or multiple mobility variables.

2.2 Memory-Less Entity Mobility Models

Although there are a wide variety of ad-hoc mobility models and different ways to classify and study them, due to their simplicity to implement and analyze Random based mobility models have proven to be the most popular and favorite of all. Random based mobility models assume that movement behaviors such as direction of travel or speed are random and unrelated with each other, often labeled as memory less mobility models.

2.2.1 Random Way-Point

Initially proposed by *Johnson and Maltz* [19], the *Random Way-Point* mobility model along with its many variants is the most well-known random-based entity mobility model. Basically, this model is made up of three random variables that control the movement of a mobile node. These are the *speed, direction* and *pause time.* The *speed* and *direction* variables dictate a node to move to a random way-point with a random speed whereas *pause time* quantify the time a node stays stationary at a given location before it again decides to move to a new way-point, i.e. change of direction and/or speed. In some literature, pause time and think time are used interchangeably. In the Random Way Point model, a mobile node will randomly choose a destination way-point and travel to that destination at a certain speed, also randomly chosen from a well-defined distribution. It will then have a pause time before it starts to move again.

Although very simple and one of the most widely used mobility model, the *Random Way-Point* model has limitations due to its decaying mean speed over time and uneven node density distribution [20].

2.2.2 Random Walk

Although random based mobility models assume the movement pattern of mobile entities to be mostly random in nature, carefully fine tuning the three random parameters will give a wide variety of other mobility models. For example, allowing

the pause time of a node to be always zero will result in the *Random Walk* Model, also known as the *Brownian Motion*. This model is used to mimic the unpredictable movement behavior of mobile nodes [18].

2.2.3 Random Direction

Another variant of the Random based model is the *Random Direction* mobility model which makes use of the speed and direction variables to determine the shape of the path a node takes. This model was proposed by *Royer, Melliar-Smith and Moser* [21] to address the problem of non-uniform spatial node distribution that result from the non-uniform movement angle distribution in Random Way Point mobility model. Therefore, instead of randomly choosing a destination way point, a mobile node with the Random Direction mobility behavior will uniformly and randomly choose direction and speed. It will then travel to the edge of the simulation area before changing its speed and direction, resulting in a uniform movement angle distribution and avoiding clustering in the middle of the simulation area.

2.3 Non-Memory-Less Entity Mobility Models

To account for some of the less-random patterns of mobile nodes, the research community have proposed a variety of models labeled as *non-memory-less* mobility models.

2.3.1 Gauss-Markov Mobility Model

The *Gauss-Markov model* assumes different degree of randomness in mobility metrics to correlate previous movement decisions with current one. It uses a single control variable, $\alpha(0 \leq \alpha \leq 1)$, to vary the randomness of movement decisions [22]. The direction of travel and speed for the next movement decision is related to the previous direction and speed by a randomness variable α. A complete randomness of the movement behavior can be achieved when $\alpha = 0$ whereas $\alpha = 1$ would result in a movement model full of memory.

2.3.2 Smooth Random Mobility Model

Originally proposed by Bettstetter [23,24], the *Smooth Random Mobility Model* tries to make movements to be more realistic and smooth by avoiding sudden and abrupt movement pattern of nodes such as sharp turn or sudden stop. For example, it assumes that velocity is node uniformly distributed rather *preferences* exist whereby a target speed among the preferences is chosen and then is adjusted incrementally to approach it.

2.3.3 Pathway Mobility Model

The *Pathway Mobility Model* is a model used to address the observation that nodes often are constrained by topological constraints such as streets and walkways, hence

moving based on paths defined on some kind of map [18].

Although nodes in the *pathway mobility model* choose their destination randomly, the direction of travel is not random. In contrast to nodes in random based models such as the *Random Waypoint model* where nodes travel to their destination freely, nodes in the *pathway mobility model* are restricted to travel only on pathways in the map resulting in a pseudo-random movement pattern.

2.3.4 Obstacle Mobility (OM) Model

During the course of travelling from its starting point to a destination, a node may encounter different obstacles that force it to change its trajectory. Such obstacles also affect radio propagation behavior or shadowing effect. The effect of an obstacle on mobility modeling has been studied for example in [25]. Hence, the *obstacle mobility model* is one good example that tries to integrate these effect into mobility modeling by placing different obstacles in the simulation area.

In the *obstacle mobility model*, the path nodes are allowed to take is constructed by first computing a *Voronoi graph* which is based on the obstacles found in that area [17]. Once the path is constructed, mobile nodes can choose to travel to a random destination using the shortest path from the pre-defined pathways. The shortest path in the *voronoi graph* can be calculated by using *Dijikstras* shortest path algorithm.

2.4 Group Mobility Models

Group mobility models are models based on the notion that nodes barely move alone and the movement pattern of a node is influenced by other nodes within the group. Next, we will briefly discuss some common *Group Mobility Models.*

2.4.1 Reference Point Group Mobility Model

In the *Reference Point Group Mobility Model* (RPGM), a group/cluster of nodes with some logical center to share mobility characteristics like speed and direction move towards their respective destination uniformly distributed within the geographical scope of the group [26]. Therefore, the RFGM model realizes the spatial dependency of nodes while maintaining individual/entity movement of nodes within the group.

2.4.2 Pursue Mobility Model

Unlike the RPGM where individual nodes within a group have their own target destination, in *Persue Mobility Model*, nodes tend to follow a common target or mimic the movement behavior of a single leader node within the group [17]. Nodes in persue mobility models may still exhibit some individual movement while following their target.

Persue Mobility Models could find applications where nodes move in a group, for example target tracking in law enforcement or a group of tourists that move together by following the tour guide.

2.4.3 Column Mobility Model

The *Column Mobility Model* represents a set of mobile nodes that share the same movement direction. Nodes may move in parallel with one another while maintaining their relative position with one another [18] and having their own reference point. In the limiting condition where a node reaches the boundary of the simulation area, the movement direction will be flipped by 180 degree ensuring the movement direction is kept the same with other nodes.

2.4.4 Nomadic Community Model

The *Nomadic Mobility Model* is yet another group mobility model where nodes in a group move following some reference point but maintain their own motion of radius of motion around a common line [17].

Unlike the *column mobility model* which is based on a similar concept and individual nodes have their own reference point, *Nomadic Community models* share a common reference point. Despite sharing a common reference point, movement in *Nomadic Community Model* is sporadic than in *Column Mobility model* where it is relatively uniform [17].

2.5 Working Day Movement Model

Mobility modeling for opportunistic networks often boils down to *Human Mobility* as nodes equipped with wireless communication capability are often carried by humans [14]. Most mobility models assume some kind of randomness in the movement behavior of humans. But studies such as [27] have shown that random based movement models do not always reflect contact patterns as seen in real traces. Moreover, we humans do not actually move in random, instead our movement behavior is influenced by a number of factors. In [28], the different factors that influence our movement model are grouped into three decision levels as the *Strategic Level*, *Tactical Level* and the *Operational Level*.

At the *Strategic level*, mobility of humans is determined by a set of daily activities. These set of activities such as going to a supermarket or buying a coffee on the way to work control movement behavior at higher level. Next comes the *Tactical level* mobility decisions to perform the set of activities set in the *strategic level* such as means of transport or route choice are made. At the third level, the *Operational level*, decisions concerning walking behavior like speed or waiting times are made. Interaction with nearby nodes or the environment is made at this level [28]. Hence, we see that the movement behavior of humans is determined by a set of complex processes.

The *Working Day Movement Model* (WDM) combines most of the known movement features and the factors influencing them at different decision levels into a

single mobility model [29]. In WDM model, movement of nodes is classified into a set of activities as *Home activity*, *Office activity*, *Evening activity with friends* that repeat every day. For achieving an objective set within some daily routine and switch between a set of daily activities, the WDM model includes *Transport model.* In this model, a set of nodes could be assigned to use different modes of transport such as *walking*, *driving* or *taking the bus.*

2.6 Summary

In this chapter, we presented and discussed different human mobility models along with common metrics deployed for evaluation of the performance of ad-hoc and opportunistic networks. Having such a wide range of ad-hoc mobility models, it is not always clear which model is suitable for what kind of network. In opportunistic urban content sharing schemes like floating content, we need to consider the microscopic movement behavior of mobile nodes, especially the patterns nodes follow while they are within the anchor zone. Mobile nodes within a city square for example tend to have a more random and memory-less movement pattern whereas group movement behavior are less relevant, especially in small sized anchor-zones.

Another consideration in the choice of mobility model is the simulation area where the assumed mobility pattern takes place. Often, mobility models assume a closed boundary of simulation area where nodes are permanent residents of the region but vary on the type of movement model they follow. Again, if we consider some cross-section of a city like square, and evaluate opportunistic networks, it is not true that this city cross section has the same set of mobile nodes. People might come and leave the region thereby varying the available number of mobile nodes in that specific region. Therefore, a number of parameters should be taken into consideration before choosing a mobility model for the study of ad-hoc networks in general.

3 Problem Description, Scope and Methodology

In this chapter, we will first present limitations of existing pedestrian mobility models for opportunistic content sharing applications in urban areas and set the main objective of this thesis. Next we will introduce different methodologies used to study pedestrian mobility models. We will discuss some of the most common approaches in conducting mobility studies by critically analyzing the pros and cons of each method, and finally select the best applicable method to studying a local pedestrian mobility model for urban content sharing.

3.1 Problem Description and Scope of this thesis

Previous works on opportunistic ad hoc content sharing and floating content has mainly concentrated on the study of the overall system performance and its ability to keep the content floating. While such "macroscopic level" evaluations gives us a good understanding of the performance of large systems (fluid limit), for small systems such as in city squares or road intersections, we need a different approach and a "microscopic level" mobility model where we capture the short term movement patterns of nodes in small areas. The basic features of opportunistic content sharing services in urban areas also require us to have an open simulation area where nodes can freely enter and leave. It is also important to have a model that captures important parameters that determine what kind of opportunistic links are formed, example by mean of modeling a mobile node's mean sojourn time. In this thesis work, we present a **square mobility model** for opportunistic content sharing in urban areas. The model tries to capture the movement of people in open city squares where people enter the square, move around for a while, and then leave.

So far, considerable efforts both analytically and through simulation have been exerted to study the concept of Floating Content [5–7]. One key result of these efforts is the criticality condition from [5], which relates the mean number of contacts during a sojourn in an anchor zone to the probability of content floating. Due to the inherent "best effort" nature of floating content, availability of a piece of content in a given location is probabilistic and highly dependent on fulfilling such criteria. But, this criticality condition is only valid at the fluid limit for which we can neglect random fluctuations and assume the average. In order to study the performance of the system at a microscopic level (for small systems such as city squares), we can no longer neglect stochastic fluctuations and therefore we need a different approach to study the performance of floating content in small systems.

Therefore, the research in this thesis project aims to contribute to the study of floating content by developing a reasonable and accurate user movement model (at the microscopic level) in locations such as city parks, hot-spots & squares, or wider regions formed by road intersections and blocks of buildings. We will also give a detailed analysis of floating content for small systems, especially when the system is initialized for the first time.

Specifically, the main objectives are:

1. To identify important parameters and their mathematical modeling of pedes-

trian mobility, applicable to opportunistic content sharing in urban areas.

2. Implementing the mobility model in simulator, and verifying results of the simulation with the analytical model.

3. Analyze the necessary conditions that increase content availability

4. Establish the conditions for a reasonable high probability of *bootstrapping*, thereby avoiding early content extinction.

3.2 Methodology

If we look deeper into most of the mobility models that are discussed in chapter 2, e.g. see also [18], we can see that all models are the derivatives of analytical models, simulation, or real life measurement. Depending on the purpose and scope of the mobility study, one or a combination of these methodologies has been used.

3.2.1 Analytical Models

A model is a simplified representation of some aspect of the real world [30]. The representation and description of a model can take different forms such as flow charts, decision trees, theoretical or conceptual models, and mathematical models. Analytical models are one form of mathematical models, as compared to numerical models, that have a closed form solution, i.e. the solution to the equations used to describe relationships and changes in a system can be expressed as a mathematical analytic function.

In studying mobility, important parameters that are constituents of a certain movement behavior are usually probabilistic in nature. For this reason, analytical models used in mobility modeling often take the form of Probabilistic (stochastic) models. These probabilistic models are based on relationships between parameters such as velocity of mobile user, pedestrian/node density, arrival rate, sojourn time, area/region of system operation, etc which are recognized as leading to a certain variable outcome (e.g. movement behavior) which however can only be probabilistically predicted.

Some of the advantages of Analytical models are:

- They are often cost-effective to study and describe essential features of user mobility patterns.

- These models can be applied to a very large scale and variety of scenarios in an efficient and comfortable manner.

- The discipline of constructing analytical models helps us to identify essential elements needed for moving towards explanation of system behavior and for communicating with others.

- In formal analytical models, a wide range of variable can be used and suppressed at the same time so that relationships among other variables can be investigated.

Despite their advantages, analytical models also have common pitfalls and limitations. One disadvantage of analytical solutions is that they are often mathematically very challenging to obtain. In some situations, they might result in a solution that is too complex to work out even numerically. Moreover, these models are usually not accurate when compared to real executions due to assumptions and simplifications in the modeling process. Real life scenarios and behaviors are functions of a wider variable set than assumed in analytical models. Therefore, analytical models often need to be accompanied by simulation tests in order to confirm all relevant characteristics are captured.

3.2.2 Simulation

The cycle of introducing new technologies and fruits of scientific researches is accelerating at an amazing rate. As a result, ideas and innovative solutions to adopt and make best use of these new technologies have become complex processes within the business, educational, political and scientific community. But not all new ideas and technologies are viable due to the existence of a number of unidentified variables. In order to identify and minimize the unknown factors governing a given system, different types of feasibility study are carried out. In the study of DTNs and challenged networks, as in many other disciplines, one form of such a study is conducted by means of simulation.

According to the definition of the Oxford Advanced Learner's dictionary, simulation is "*a situation in which a particular set of conditions is created artificially in order to study or experience something that could exist in reality*" [30]. Simulation has proven to be one of the most indispensable tools in the study of DTN networks. This is because practical (real life) tests of applications, protocols and system models are not economically feasible, especially when they are at their early stage of development. Additionally, not every model, protocol or system performance can be fully understood by the use of theoretical and mathematical formulations.

Mobility modeling for opportunistic networks is no different. This is because probabilistic (stochastic) user mobility modeling takes a number of assumptions and simplifications in order to formulate a closed form solution. But these assumptions and simplifications should be taken carefully not to neglect the important characters out of the model and compromise quality. Therefore, simulation of user mobility will ensure the quality of the model we obtain.

3.2.3 Real life Trace-based Models

Although analytical or simulation based mobility models are most frequently used models in the study of ad-hoc type networks and applications, such synthetic mobility models do not always capture real user movement behaviours. This is because

most models assume basic user movement activities, for example simple random-walk models and their variations which are often limited by the imagination of their designers. Therefore, it has become a common practice to supplement such models by using real-world user movement traces. A careful characterization and analysis of user mobility traces will result in better understanding of the different factors involved to formulate certain movement behaviour.

Mobility research based on real-life measurements and dataset are made because of their realistic significance. For example, a Community Resource for Archiving Wireless Data At Dartmouth (CRAWDAD) serves as a public archive and provides the research community with a wide range of wireless trace data. These data can be used by researchers in various DTN type networks and applications. Recent works on opportunistic type ad-hoc networks, such as [31], have also contributed towards this growing data set. Although the methodology used in [31] provides real contact opportunities, it also has its own limitations as it is based on small populations. Other data sets consider larger populations such as [32] and [33] at Dartmouth College and [34] at UCSD. Yet again, these datasets do not capture all contact opportunities, rather only when users under the same Wi-Fi access point (AP) come in range [35].

Therefore, if and when available, real-world traces can be used to better understand and formulate realistic mobility models. One can also set up own experiment and collect mobility traces from usage statistics such as `tcpdump` and *syslog*, for example as in [36].

3.3 Summary

In this chapter, we presented three different methodologies used to craft a mobility model for opportunistic ad-hoc networks. Considering the pros and cons of the different methodologies we have discussed, it is evident that using a combination of analytical modeling and simulation tests will ensure the quality of the model we obtain and that the objectives of this thesis work are met. It is also important that mobility modeling of users should take a balance between being too simple or sophisticated, and capturing all or part of the important characteristics of a mobile user for the service in question. Therefore, we will first analyze pedestrian mobility from a mathematical point of view and derive a probabilistic (stochastic) model for opportunistic content sharing in urban areas. Next, we will implement the model in a simulator. Finally we will run a number of simulation tests in order to ensure all important parameters of user mobility are captured and reflected by the model.

4 A Local Pedestrian Mobility Model for Urban Content Sharing

In order to fully understand and analyze the capability of opportunistic content sharing networks such as Floating Content, various types of feasibility studies and simulations are required. Among the different components, the mobility of pedestrians in a given location is one of the critical design aspects which need to be thoroughly examined and modeled.

In this chapter, we will introduce a specific mobility model for pedestrians in open city squares or parks. The floating content concept can be envisioned to be utilized in such places as there is typically a constant flow of people, squares have monuments and other tourist attractions, and e.g. interesting shops can be found nearby.

4.1 The Square Mobility Model

Let us consider a rectangular anchor zone depicted in Figure 3, bounded by roads and buildings so that the only entry (and exit) points are located at the four corners of the zone. For intra zone movement we will adapt the well-known Random Way-Point (RWP) model, which has been studied extensively in past, see e.g. [37–41].

Formally, the *square mobility model* is defined follows.

- Nodes arrive according to a Poisson process to four corners of the square, from where they travel immediately to a random way-point in the anchor zone. Velocity is some constant v.
- Upon reaching a way-point in the square, a node throws a dice. With probability of $P^{(\text{exit})}$, the node decides to leave the area and chooses the exit corner again uniformly in random. Otherwise node chooses a next way-point uniformly in random in the square. In either case, the node moves to the next destination with velocity of v.

Consequently, each visit in the anchor zone comprises two legs from/to a corner to/from a random way-point, and X RWP style legs within the square, where X is a geometrically distributed random variable with parameter $P^{(\text{exit})}$, i.e.,

$$\mathrm{E}[X] = \frac{1 - P^{(\text{exit})}}{P^{(\text{exit})}}.$$

We note that squares with tourist attractions may have a lower exit probability $P^{(\text{exit})}$ than squares where pedestrian mainly pass through the region. Furthermore, we let $\mathrm{E}[L_e]$ denote the mean leg length between a corner and a random way-point, and $\mathrm{E}[L_{\text{rwp}}]$ the mean length of RWP leg.

The square mobility model is based on the $M/G/\infty$ queuing system where nodes arrive/enter the anchor zone according to the Poisson process with intensity λ. The nodes are served by infinite servers, each of whom has an arbitrary service time distribution G.

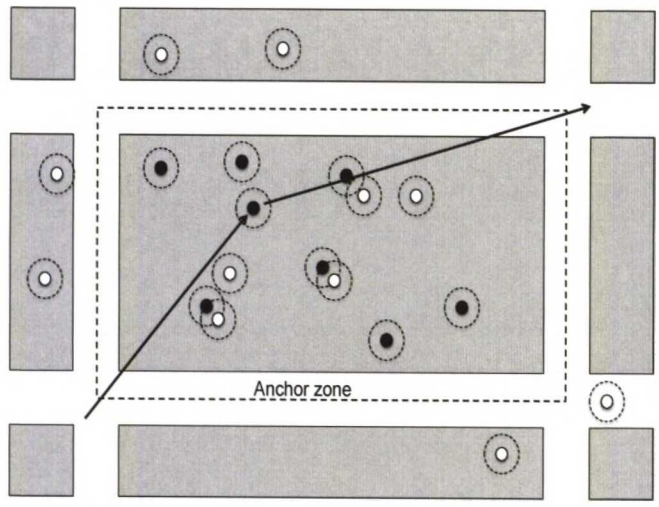


Figure 3: A rectangular anchor zone with a node entering at the bottom left, changing direction once, and then exiting at the top right corner. "Black" nodes carry a copy of the content (only inside the anchor zone), white ones do not.

In order to measure and analyze the performance of floating content based on the criticality condition in (1) and the square mobility model, the two most important parameters are the mean sojourn time, and contact rate. In the floating content application, a mobile node entering the anchor zone will replicate content as long as it stays and meet other nodes within the anchor zone. Therefore, the mean sojourn time, the average time a mobile node is expected to spend in the anchor zone before leaving is an important parameter. Obviously, the longer a node stays in the region, the more the probability of content replication and content availability. Similarly, the mean contact rate of a node during its sojourn time is important as replication of content occurs only when there is contact. Here we define contact rate as the number of contacts a node makes per unit time while it is roaming in the anchor zone.

4.2 Mean Path Length and Sojourn Time

Let us next consider the rectangular area with side lengths $a \times b$ as depicted in Fig. (4). Without loss of generality, we can assume that $a \geq b$. Let c denote the diagonal of the rectangle, $c{=}\sqrt{a^2 + b^2}$, and r the random distance a node travels from the entry corner to the first way-point.

If we consider equi-distant points from one corner of the rectangle, we can see that these points actually form a tiny arc bounded by sides of the rectangle with the vertex being center of the arc. We can also see that weight(number of these points) correspond to the arc length which is directly dependent on the radius "r" of this tiny arc. But the length of the arc formed by these points vary from being a quarter of a circle for $r{\leq}b$ up to a single point for $r{=}c$. Next consider three cases: $r \leq b$, $b < r \leq a$ and $a < r \leq c$

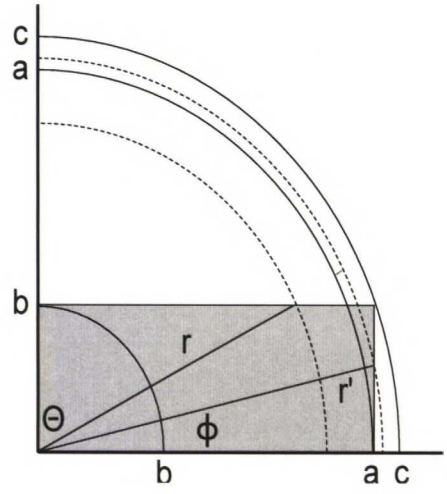


Figure 4: Rectangular anchor zone with side lengths a & b.

Case 1) For $r \leq b$ we have the distance distribution as a function of r, which is also one quarter of the perimeter of a circle with radius r. Hence,

$$f(r) = \frac{2\pi r}{4}, \; 0 \leq r \leq b \tag{2}$$

Case 2) For $b \leq r \leq a$ we have the distribution as one quarter of the perimeter of a circle with radius r minus the arc length which falls outside of the rectangle. From figure (2) we can see that the subtending angle forming the arc which falls outside the rectangle is $\theta = \arccos(\frac{b}{r})$. Hence,

$$f(r) = \frac{\pi r}{2} - r\theta, \quad b \leq r \leq a. \tag{3}$$

Case 3) For $a \leq r \leq c$ we have the distribution as one quarter of the perimeter of a circle with radius r minus the arc length which falls outside of the rectangle. But for this case apart from the arc formed by θ, we have one more arc falling outside the rectangle area with subtending angle $\phi = \arccos(\frac{a}{r})$. Hence,

$$f(r) = \frac{\pi r}{2} - r\theta - r\phi, \text{ if } a \leq r \leq c \tag{4}$$

Combining equations (2)-(4) gives the distance distribution:

$$f(r) = r \begin{cases} \frac{\pi}{2}, & 0 \leq r \leq b, \\ \frac{\pi}{2} - \arccos(\frac{b}{r}), & b \leq r \leq a, \\ \frac{\pi}{2} - \arccos(\frac{b}{r}) - \arccos(\frac{a}{r}), & a \leq r \leq c. \end{cases} \tag{5}$$

Now that we have the distance distribution in a rectangular area, we can calculate the expected distance from/towards one of the corners of the rectangle by the

formula:

$$\mathrm{E}[L_e] = \frac{\int_0^c r f(r)\,\mathrm{d}r}{\int_0^c f(r)\,\mathrm{d}r}. \tag{6}$$

Equation (5) gives $r\,f(r)$, and hence we can write the numerator of (6) as:

$$\begin{aligned}\int_0^c r f(r) dr = \frac{ab\sqrt{a^2+b^2}}{3} + \frac{a^3}{6}\ln\left(\frac{b+\sqrt{a^2+b^2}}{a}\right) \\ + \frac{b^3}{6}\ln\left(\frac{a+\sqrt{a^2+b^2}}{b}\right)\end{aligned} \tag{7}$$

whereas, solving the denominator of equation (6) gives us:

$$\int_0^c f(r)\,\mathrm{d}r \;= a{\times}b \text{ which is the area of the rectangle.} \tag{8}$$

Substituting (7) and (8) into (6) gives $\mathrm{E}[L_e]$,

$$\mathrm{E}[L_e] = \frac{c}{3} + \frac{a^2}{6b}\ln\left(\frac{b+c}{a}\right) + \frac{b^2}{6a}\ln\left(\frac{a+c}{b}\right). \tag{9}$$

Equation (9) gives the average length of a node's movement during the entry/exit of the anchor zone. The expected epoch length for the node's RWP movement within a rectangular anchor zone of sides $a \times b$ is given in [37] which reads:

$$\begin{aligned}\mathrm{E}[L_{\mathrm{rwp}}] = \frac{1}{15}\left[\frac{a^3}{b^2} + \frac{b^3}{a^2} + c\left(3 - \frac{a^2}{b^2} - \frac{b^2}{a^2}\right)\right] \\ + \frac{1}{6}\left[\frac{b^2}{a}\operatorname{arcosh}\frac{c}{b} + \frac{a^2}{b}\operatorname{arcosh}\frac{c}{a}\right]\end{aligned} \tag{10}$$

Finally, the *average distance a node travels* within the anchor zone reads,

$$\mathrm{E}[L] = 2\mathrm{E}[L_e] + \frac{1 - P_{\mathrm{exit}}}{P_{\mathrm{exit}}}\mathrm{E}[L_{\mathrm{rwp}}], \tag{11}$$

where $\mathrm{E}[L_e]$ and $\mathrm{E}[L_{\mathrm{rwp}}]$ are given by (9) and (10), respectively. The mean distance a node travels in the anchor zone is an important quantity as, e.g., together with the velocity v and arrival rate λ, it defines the mean number of nodes in the anchor zone (cf. Little's result),

$$N = \lambda\,\mathrm{E}[L]/v, \tag{12}$$

where velocity v were some constant. For a random leg or visit specific velocity the above takes form $\lambda\,\mathrm{E}[L]\,\mathrm{E}[1/v]$.

4.3 Contact Rate

The total contact rate, denoted by R, is the total rate at which nodes meet in the anchor zone. It is one important quantity when assessing the feasibility of opportunistic schemes such as the floating content. The total contact rate can be computed exactly for the square mobility model using the formulæ given in [5]. For simplicity, here we simply estimate it by assuming that the movement pattern in the anchor zone were isotropic, i.e. all directions are equally likely. Additionally, we assume point contacts, i.e., typical transitions are much longer than the transmission range d. Two nodes are said to have encountered each other when the distance between them becomes less than the transmission range d. This elementary model for radio propagation is known as Gilbert's model [42].

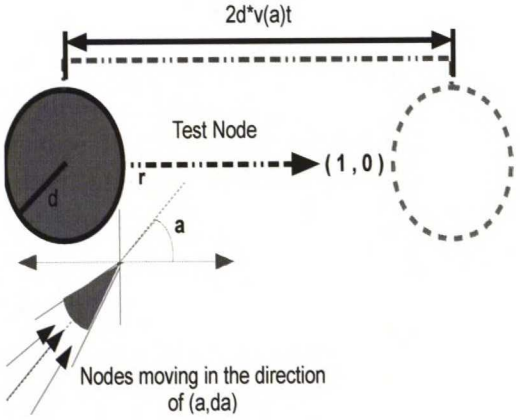


Figure 5: Illustration of contact rate for a test node at r and moving in the direction of $(1, 0)$ along the x-axis and nodes moving in the direction of $(a.da)$.

Let us assume a uniform node distribution within the anchor zone and denote by ρ, the spatial node density such that $\rho = N/A$. Next, if we consider nodes which move in the direction of $(a, a + da)$, figure (5), such that a=$(0, 2\pi)$, their spatial density will be:

$$\tilde{\rho} = \rho \frac{da}{2\pi} \tag{13}$$

Now for a test node at r and travelling in the direction of $(1, 0)$ along the x axis, its relative velocity with the nodes moving in the direction of a will take the form:

$$v(a) = v\sqrt{(1 - \cos^2 a) + (\sin^2 a)} \tag{14}$$

For a node with a transmission range of d, its actual encounter region can be modeled as a circular area of radius d. During a certain time period t, a test node at r will span an effective encounter area of $A_{enc} = 2d{\cdot}v(a)t$ having a total of $2d{\cdot}v(a)t{\cdot}\tilde{\rho}$ encounters with nodes travelling in the direction of a. Substituting equation (13) for $\tilde{\rho}$ and (14) for $v(a)$, the total contact rate with these nodes will be:

$$\tilde{R} = 2dv(a)\rho\frac{\mathrm{d}a}{2\pi} = \frac{\sqrt{(1-\cos^2 a) + (\sin^2 a)}}{\pi}\rho dv \cdot \mathrm{d}a$$

Integrating the above equation over all possible angles, i.e.,a=$(0, 2\pi)$ will give us the contact rate per unit time of the test node, R_i, which reads:

$$R_t = \frac{8}{\pi}\rho dv$$

and,

$$R = \frac{N \cdot R_t}{2} = \frac{4}{\pi}\frac{dvN^2}{A}. \tag{15}$$

Hence, equation (15) gives the total contact rate (or encounter rate) within a rectangular shaped anchor zone of area A.

4.4 Stationary Spatial Node Density

In this section, we will derive an expression for the stationary node distribution of the *Square Mobility Model*. For that, let us first drive a general expression for a general mobility model where:

- Nodes move in a convex area $\mathcal{A}$.
- New nodes arrive at m_{in} locations $\{\mathbf{r}_i^{(\text{in})}\}$ with rates $\{\lambda_i\}$, $i = 1, \ldots, m_{\text{in}}$. (i.e., the system has m_{in} sources)
- Upon arrival, a node chooses its first waypoint uniformly in random from $\mathcal{A}$ and starts moving directly towards it.
- Upon reaching the waypoint j:
  - The node keeps an i.i.d. random pause time $\tau_j \sim \tau$, with $\mathrm{E}[\tau] < \infty$.
  - With probability of $p_i^{(\text{exit})}$, the node moves next to the exit point $\mathbf{r}_i^{(\text{out})}$, $i = 1, \ldots, m_{\text{out}}$, where it then departs the system (i.e., there are m_{out} sinks).
  - Otherwise, the node chooses a new waypoint again uniformly in random from $\mathcal{A}$ and starts moving directly towards it.
- Velocity v for each transition is chosen independently from an arbitrary velocity distribution with $\mathrm{E}[1/v] < \infty$

That is, in the above model nodes arrive at certain fixed locations, then move according to RWP for a random number of transitions, and then depart the system in one of the fixed exits. We let λ denote the total arrival rate,

$$\lambda = \sum_{i=1}^{m_{\text{in}}} \lambda_i,$$

and $P^{(\text{exit})}$ the probability of exiting after completing a leg,

$$P^{(\text{exit})} = \sum_{i=1}^{m_{\text{out}}} p_i^{(\text{exit})}.$$

Necessary conditions for a stable system are that $P^{(\text{exit})} > 0$ and that $\mathrm{E}[1/v] < \infty$, which ensure a finite mean sojourn time.

The stationary node density comprises three types of components:

(i) transitions to/from a source/sink,

(ii) RWP-style transitions between two uniformly distributed random waypoints in $\mathcal{A}$, and

(iii) pause times occurring uniformly in $\mathcal{A}$.

Consider next a sample node arriving to source i located at $\mathbf{r}_i^{(\text{in})}$ as illustrated in Fig. 6. The sample node moves through a differential area da at $\mathbf{r} = (x, y)$, if its first waypoint is somewhere in dA. For the differential areas we have,

$$da = (a_1 + r) \cdot d\theta \cdot dr,$$
$$dA = a_1 \cdot d\theta \cdot dh.$$

The arrival rate of such nodes is:

$$\lambda_i \cdot \frac{dA}{A},$$

where A denotes the area of $\mathcal{A}$. All these nodes spent time $dh \cdot \mathrm{E}[1/v]$ on average inside da, and according to Little's result [43], the mean number of them in da is:

$$\lambda_i \cdot \frac{dA}{A} \cdot dh \cdot \mathrm{E}[1/v].$$

Therefore, their contribution to the node density at da is:

$$dn_i = \frac{\lambda_i \cdot dA \cdot dh \cdot \mathrm{E}[1/v]}{A \cdot da} = \frac{\lambda_i (a_1 + r)\,\mathrm{E}[1/v]}{A a_1}\, dr, \tag{16}$$

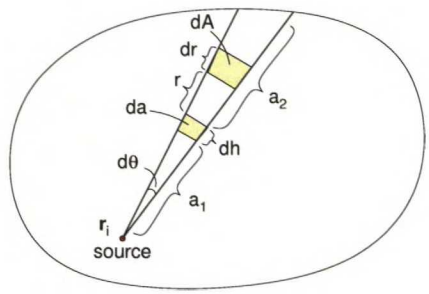


Figure 6: Derivation of the node density of enter/exit legs.

and integrating the above, (16), over r gives

$$n_i(\mathbf{r}) = \frac{\lambda_i \,\mathrm{E}[1/v](2 + a_2/a_1)a_2}{2A}. \tag{17}$$

The departures are similar to arrivals and essentially (17) holds. The rate at which nodes depart through sink i at $\mathbf{r}_i^{(\mathrm{out})}$ is $\lambda_i^* = \lambda \cdot p_i^{(\mathrm{exit})}/P^{(\mathrm{exit})}$, and we have

$$n_i^*(\mathbf{r}) = \frac{\lambda_i^* \,\mathrm{E}[1/v](2 + a_2/a_1)a_2}{2A}. \tag{18}$$

Let k denote the mean number of the RWP transitions a node takes before exiting, such that:

$$k = \frac{1 - P^{(\mathrm{exit})}}{P^{(\mathrm{exit})}}.$$

The rate at which nodes reach waypoints is $\lambda(k+1)$, which includes also the first (unconditional) waypoint. The density of nodes having a pause is a constant across $\mathcal{A}$,

$$n_{\mathrm{p}}(\mathbf{r}) = \lambda(k+1)\,\mathrm{E}[\tau]/A. \tag{19}$$

Similarly, RWP-style transitions occur at rate λk, each lasting time $L_{rwp}\,\mathrm{E}[1/v]$ on average, where L_{rwp} denotes the mean length of RWP leg. With the aid of Little's result, the mean number of nodes moving according to RWP is,

$$N_{rwp} = \lambda\, k\, \mathrm{E}[1/v]\, L_{rwp}.$$

The stationary node density for RWP in an arbitrary convex area has been derived in [40] and it reads,

$$f_{rwp}(\mathbf{r}) = \frac{1}{A^2 L_{rwp}} \int_0^{\pi} a_1 a_2 (a_1 + a_2)\, d\theta,$$

where L_{rwp} denotes the mean leg length, and a_1 and a_2 are the distances to the boundary from $\mathbf{r}$ in directions θ and $\theta + \pi$, respectively. Therefore, in our model, the density of nodes concurrently moving on RWP legs is $N_{rwp}\, f_{rwp}(\mathbf{r})$, which then gives:

$$n_{rwp}(\mathbf{r}) = \frac{\lambda\, k\, \mathrm{E}[1/v]}{A^2} \int_0^{\pi} a_1 a_2 (a_1 + a_2)\, d\theta. \tag{20}$$

The total density of nodes is the sum of the four components

$$n(\mathbf{r}) = \sum_{i=1}^{m_{\mathrm{in}}} n_i(\mathbf{r}) + \sum_{i=1}^{m_{\mathrm{out}}} n_i^*(\mathbf{r}) + n_{rwp}(\mathbf{r}) + n_{\mathrm{p}}. \tag{21}$$

where the first two terms of the expression in the right are given by equations (17) and (18) whereas the last two terms by (20) and (19).

4.4.1 Square mobility model

The square mobility model is a special case of the general mobility model described above: $\mathcal{A}$ is the $a \times b$ rectangle with $m_{\text{in}} = m_{\text{out}} = 4$ co-located sources and sinks, $\mathbf{r}_i^{(\text{in})} = \mathbf{r}_i^{(\text{out})}$, having uniform arrival rates, $\lambda_i = \lambda/4$, and uniform exit probabilities, $p_i^{(\text{exit})} = P^{(\text{exit})}/4$. Furthermore, the pause times are zero, $\tau = 0$, and the velocity a constant v.

The mean number of nodes in the area is given by (12):

$$N = \lambda(2\,L_e + k\,L_{rwp})/v,$$

where L_e denotes the mean distance from a corner to a random point in the area.

The situation for the square mobility model is illustrated in Fig. 7, where the origin is at the lower left corner. Due to the symmetry (i) in entering and exiting, and (ii) in rectangular topology, the density of nodes entering or exiting the area can be written as:

$$n_e = n_1(x, y) + n_1(a - x, y) + n_1(x, b - y) + n_1(a - x, b - y),$$

where $n_1 = n_1(\mathbf{r})$ denotes the contribution of nodes entering and/or exiting the area from corner 1, for which (17) gives,

$$n_1(x, y) = \frac{\lambda}{4abv}\left(2 + \frac{a_2}{a_1}\right) a_2,$$

where

$$a_1 = \sqrt{x^2 + y^2},$$
$$a_2 = \sqrt{x^2 + y^2}(\min\{a/x, b/y\} - 1).$$

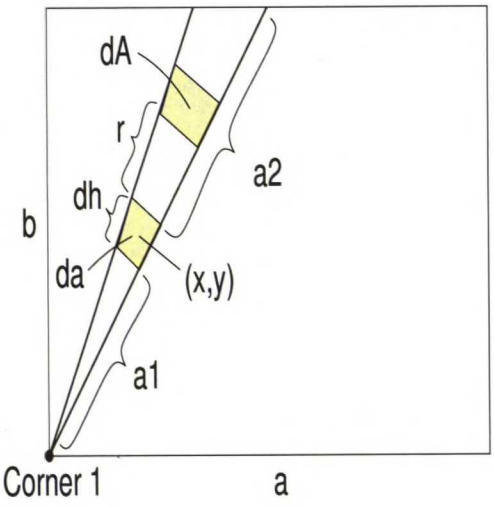


Figure 7: Node density of enter/exit legs for square mobility model.

As an example, let us consider the unit square, $a = b = 1$. In this case,

$$\begin{aligned} L_{rwp} &= (1/15)(2+\sqrt{2}+5\,\ln(1+\sqrt{2})) && \approx 0.521, \\ L_e &= (1/3)(\sqrt{2}+\ln(1+\sqrt{2})) && \approx 0.765, \end{aligned}$$

so that the mean number of nodes in the area is

$$\begin{aligned} N &= \frac{\lambda}{15v}\left(10\sqrt{2}+10\ln(1+\sqrt{2})+k(2+\sqrt{2}+5\ln(1+\sqrt{2}))\right) \\ &\approx \frac{\lambda}{v}\left(0.521\cdot k+1.53\right). \end{aligned}$$

The stationary node distribution is illustrated in Fig. 8 for $k = 0$(top-left), $k = 2$(top-right), $k = 10$(bottom-left), $k = \infty$(bottom-right) using (21). With $k = 0$, each node leaves the area immediately and the node distribution has remarkably sharp ridges along the diagonals. Moreover, due to the finite set of four enter/exit points at the corners, the density function has a singularity at the corners.

Case $k = \infty$ corresponds to limit when $\lambda \to 0$ and $k \to \infty$ at the same time in order to have a stable population. In this case, the node distribution is according to the RWP. We observe that the node density is zero in the boundary, whereas in the case of $k = 0$, the node density is strictly positive also in the boundary (and even infinite at the corners). By varying the parameter k (i.e., the exit probability P_{exit}), the stationary node distribution changes smoothly between the two extremes. In all cases, a local maximum is at the center of the area.

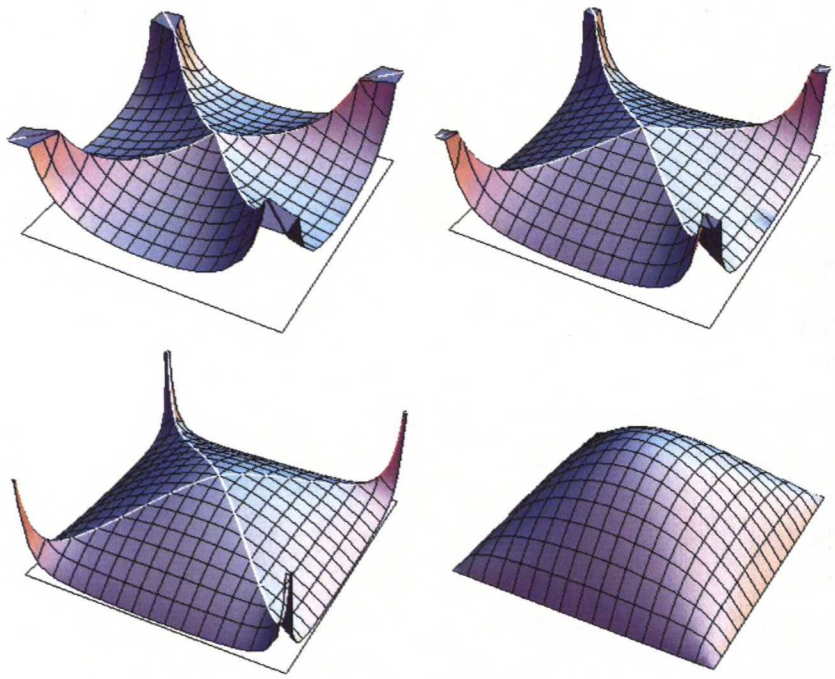


Figure 8: Stationary spatial distribution of nodes according to the square mobility model in unit square with the mean number of RWP legs $k = 0, 2, 10$ and ∞ (i.e., $k = (1 - P^{(\text{exit})})/P^{(\text{exit})}$).

4.5 Summary

In this chapter, we presented our pedestrian mobility model, the *Square Mobility model*, for opportunistic content sharing applications in urban areas such as city squares and parks. We defined and analyzed a model for capturing the movement of people in open city squares where people enter the square, move around for a while and then leave.

We started by defining the *Square Mobility Model* using only one free parameter, $P^{(\text{exit})}$, which controls the shape of the path a node takes in the square before departing. We have captured some of the most important parameters of the model such as the *mean path length* and *sojourn time*, *total encounter rate* within the square, and the *stationary spatial node distribution* and derived sound mathematical expressions for their evaluations. Notably, we derived a closed form expression for evaluation of the *mean path length* in the square and found out that when $P^{(\text{exit})} \to 0$, the *spatial node distribution* marginally approaches that of RWP and when $P^{(\text{exit})} = 1$, nodes simply make a stop in the square and then depart without making any further RWP-style movement.

5 Floating Content Analysis

In this chapter, we will present analysis of floating content in both spatial and non-spatial domains. In the first two sections, we will consider floating content in a non-spatial black-box model and assess information availability and bootstrapping of the system with small number of nodes. In section 5.3, we will consider spatial aspects of floating content with the **Square Mobility** model and study how it performs under this model.

5.1 Information availability

We will study a non-spatial model for the number of information carrying nodes given in [5]. In this system, the number of nodes in the anchor zone, denoted by N, is assumed to be so large that random fluctuations in N can be neglected. Moreover, the mean sojourn time of nodes is $1/\mu$ and ν denotes the frequency at which a pair of nodes come in contact with each other. For simplicity, the system is further assumed to constitute a Markov process. Letting p denote the fraction of nodes carrying the content, the net growth rate for $p = p(t)$ satisfies the following differential equation [5]:

$$N\frac{dp}{dt} = N^2 p(1-p)\nu - Np\mu. \tag{22}$$

The first term on the right-hand side corresponds to the rate at which content is replicated and the second term to the rate at which information carrying nodes leave the anchor-zone. When dp/dt is negative, the content tends to sink and when dp/dt is positive the fraction of information carrying nodes tends to increase. At a steady state dp/dt is zero.

Let us first establish the necessary condition for reaching penetration p higher than $p^* > 0$ in steady state. At the steady state $dp/dt = 0$ and (22) has one positive solution:

$$\frac{N\nu}{\mu} > \frac{1}{1-p^*}, \tag{23}$$

which gives (1) when $p^* \to 0$. As long as the above inequality holds, the system has a steady state where the fraction of nodes having the content is higher than p^*.

Conversely, we have:

$$p < 1 - \frac{\mu}{N\nu} = 1 - \frac{\mu^2}{\lambda\nu}, \tag{24}$$

where $\frac{N\nu}{\mu} = m$ is the *mean number of contacts per visit.* Hence, we can write (24) as:

$$m > \frac{1}{1-p} \tag{25}$$

Figure (9) depicts the fraction p at the steady state as a function of m. At the criticality threshold $m \to 1$ (from above), i.e., each node meets only one node on average before leaving the anchor-zone. That is, $p \to 0$ and most nodes visiting the anchor zone cannot obtain the content even though it does exist somewhere there.

Hence, in order to achieve a reasonable *availability*, as stated in Section 3, we require that the fraction of nodes having the content is strictly positive, $p > 0$. From the figure, we see that in order to achieve a higher content penetration, the system needs to operate clearly above the criticality condition.

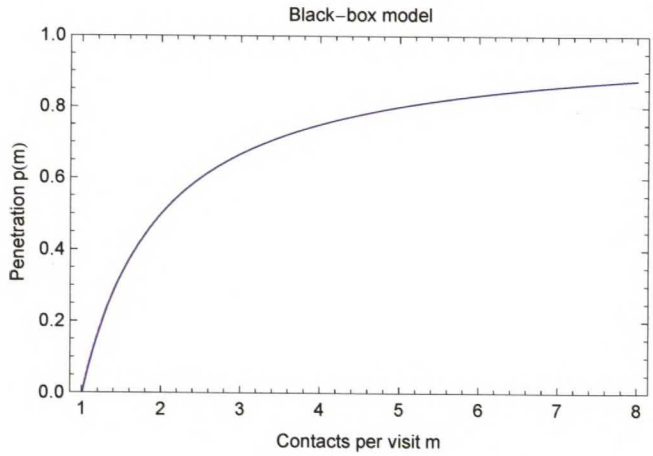


Figure 9: Mean penetration p according to (25)

For example, in order to have p=0.75, the mean number of contacts per visit must be higher than 4,

$$m > 4,$$

which is four times higher than what sustaining the content in the area requires (i.e., the criticality condition (1)).

The number of information carrying nodes a random node visiting the anchor zone meets (at the steady state) obeys Poisson distribution with mean $mp = m - 1$, where $m \geq 1$. Let z denote the probability that a node acquires the content. The necessary condition for *acquiring information with probability of* z then reads

$$m \geq 1 - \ln(1 - z) \tag{26}$$

For example, $z = 0.95$ gives $m > 3.996 \approx 4$, which corresponded to 75% penetration (see above).

Similarly, the auctors of [7] have indicated that for increased content availability and longer floating duration, the system operation point should be well above the minimum criticality condition from (1). The relationship between floating duration (probability of content floating for a certain duration) and the criticality factor presented in [7] shows that when the criticality factor m is close to 10, the probability of floating content until its intended lifetime expires becomes almost certain. Equation (25) and (26) quantify these relationships where $m > 10$ as a 90% *content penetration* and 99.99% *availability* explaining the more stringent findings of [7].

Hence, together with (26), the condition in equation (25) dictate how much above the criticality threshold of (1) should the system operate to meet primary floating objectives such as *content penetration* or *content availability.*

5.2 Bootstrapping Analysis

In this section, we will investigate the survivability of floating content during the initial transient stages, where due to random fluctuations, there is a high chance of content extinction (failure). When the system is considered to be relatively small, e.g., N=5 concurrent nodes in the anchor zone, information availability in the region is prone to stochastic fluctuations and a given piece of content can suddenly become extinct if and when p=0. Once the system has reached this state, it is no longer possible to revive the corresponding content. As nodes are always entering and leaving the anchor zone, we can consider the system to be always in transient mode from one state to other where the number of content carrying nodes fluctuate.

To this end, let us consider the Moran model [44] from Biology which describes the evolution of a given allele of a gene within a constant population size of N. If we consider the population to be closed, i.e. the system is modeled as birth-death process strictly within the population, then the proportion of the corresponding allele in the population will grow, stay the same or diminish and become extinct. Once all the genes in the population acquire an allele, the population will no longer become free of that allele. At the same time, if all genes with the given allele die, then the population will be totally free and stay free. In Moran model, these two states are the absorption states. We can apply similar analogy to study the initial operation of floating content where once all nodes having a piece of content exit the anchor zone, the system will fail (content becomes extinct). Conversely, if all nodes acquire the content, the realization is considered to be successful.

5.2.1 Probability of Content Absorption

As we have discussed earlier, the floating content is viable only when there is a sufficient number of nodes in the anchor-zone, and at least one them has the corresponding content. However, upon the content creation, only a single node has it and the system is prone to an early extinction due to the stochastic fluctuations.

We can model this bootstrapping phase by a *continuous time Markov birth-death process* with N+1 states where N is the average number of nodes in the anchor zone and state S_i is the state with i nodes having a copy of the content. The absorbing state S_0, where the content has *sunk*, is considered as a *failure*, and state S_N as a *success*. The replication of content corresponds to a birth (state transition $S_i \rightarrow S_{i+1}$), and an information carrying node leaving the anchor-zone corresponds to a death ($S_i \rightarrow S_{i-1}$).

Let β_i denote the rate at which the number of information carrying nodes grows in state S_i, and α_i the rate at which it decreases. Let U_i denote the probability of absorption into state S_0 when system is initially in state S_i (see figure 5). The probabilities of state transitions $S_i \rightarrow S_{i+1}$ and $S_i \rightarrow S_{i-1}$ are denoted by $P_{i,i+1}$ and $P_{i,i-1}$, respectively:

$$P_{i,i+1} = \frac{\beta_i}{\beta_i + \alpha_i}, \quad \text{and} \quad P_{i,i-1} = \frac{\alpha_i}{\beta_i + \alpha_i}$$

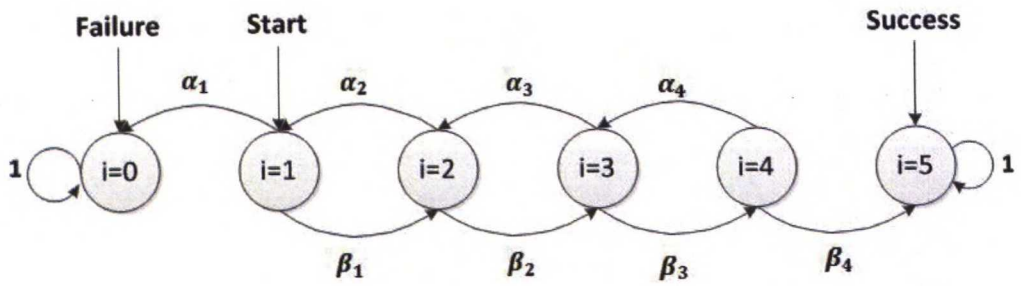


Figure 10: Floating Content as Birth-Death Process

Applying the first step analysis, U_i can be written as:

$$U_i = \frac{\beta_i}{\beta_i + \alpha_i} U_{i+1} + \frac{\alpha_i}{\beta_i + \alpha_i} U_{i-1},$$

where $i \geq 1$ and $U_0 = 1$.

As we are interested in the probability of absorption when the system is initialized, we will solve for U_1. After performing some mathematical manipulations, one obtains,

$$U_1 = \frac{\sum_{i=1}^{N-1} \rho_i}{1 + \sum_{i=1}^{N-1} \rho_i}, \tag{27}$$

where ρ_i is given by:

$$\rho_i = \frac{\alpha_1 \alpha_2 ... \alpha_{i-1} \alpha_i}{\beta_1 \beta_2 ... \beta_{i-1} \beta_i} = \prod_{k=1}^{i} (\frac{\alpha_k}{\beta_k}). \tag{28}$$

From [5], we know that,

$$\beta_i = N^2 p_i (1 - p_i) \nu, \quad \text{and} \quad \alpha_i = N p_i \mu,$$

where p_i is the proportion of information carrying nodes at state S_i, i.e., p_i=i/N, giving

$$\beta_i = (Ni - i^2)\nu, \quad \text{and} \quad \alpha_i = i\mu.$$

Hence, α_i/β_i=$(\mu/\nu)/(N - i)$ and substituting this to (28) gives

$$\rho_i = \prod_{k=1}^{i} \frac{\mu}{\nu} \frac{1}{N-k} = \left(\frac{\mu}{\nu}\right)^i \frac{(N-1-i)!}{(N-1)!}. \tag{29}$$

which concludes our derivation for *probability of content absorption* when the system is initialized.

5.2.2 Example

Consider a small scenario illustrated in Fig. (10) with only N=5 nodes in the anchor zone. Substituting (29) with $N = 5$ to (27) gives

$$U_1 = 1 - \frac{24}{24 + z(6 + z(2 + z + z^2))},$$

where $z = \mu/\nu$.

If the system operates at the criticality threshold, i.e., the derivative of (22) is zero, we have μ=4ν, and the above gives

$$U_1 = 47/50 = 0.94,$$

i.e., a system operating at the criticality threshold has a very high failure probability.

Suppose next that μ=ν. In this case, (22) has a positive derivative indicating a net growth in content density, and we obtain

$$U_1 = 5/17 \approx 0.29,$$

which is a much more reasonable result.

We see that at initial transient stage, e.g. when the system is initialized for the first time, there is an increased danger of content extinction due to stochastic fluctuations. To reduce this danger, successful bootstrapping is essential which can be achieved when state transitions $S_i \rightarrow S_{i+1}$ occurs. Note that *absorption probability* decrease whenever state transitions $S_i \rightarrow S_{i+1}$ occur, and *probability of success* increases for the same. One means of ensuring such transitions occur is if the original seeder node stays in the anchor zone long enough. But due to the inherent nature of content floating, the system cannot enforce the seeder node to stay beyond its will. Therefore, the above simple examples with a *small system size* suggest that in order to prevent the content from sinking early (with high probability) and also bootstrap the system successfully, one should be operating clearly above the criticality threshold.

5.3 Floating Content in City Square

In this section, we will analyze floating content in city squares using the square mobility model. For that, we will take three hypothetical squares of size 100 m × 100 m, 250 m × 250 m, and 500 m × 500 m and three real-world squares namely *Plaza Mayor* found in Madrid, Spain, *Place Bellecour* from Lyon, France and *Piazza Della Republica* found in Florence, Italy. The size of these squares are approximately 129 m×94 m, 300 m×200 m and 74 m×74 m respectively[3]. Although exit probability in real-world squares is characterized by different factors such as weather condition, presence of monuments and tourist attractions, in our analysis we don't consider these factors to contribute towards our shape parameter $P^{(\text{exit})}$. Instead here we will simply calculate $P^{(\text{exit})}$ as the ratio of the "total exit area" formed by e.g. roads to the total area of the square.

Let us first consider one of the above square anchor zone with side lengths a=b=100 m and exit probability P_{exit}=0.436 [4]. The expected distance for the node's entry or exit is given by (9) as:

[3]Approximate measurement of squares can be computed from Google Maps, https://maps.google.com/, and associated distance measurement tools.

[4]Double lane roads with typical width of 3.7 m and a pedestrian side-walk of 2.8 m would result in a 13 m wide exit area wrapped around the square.

$$\begin{aligned} \mathrm{E}[L_e] &= \frac{c}{3} + \frac{a^2}{6b}\ln(\frac{b+c}{a}) + \frac{b^2}{6a}\ln(\frac{a+c}{b}) \\ &= 56.6245m \end{aligned} \tag{30}$$

whereas epoch length for the random way-point movement within the anchor zone is given by (10) as:

$$\begin{aligned} \mathrm{E}[L_{\mathrm{rwp}}] = & \frac{1}{15}[\frac{a^3}{b^2}+\frac{b^3}{a^2}+c(3-\frac{a^2}{b^2}-\frac{b^2}{a^2})] \\ & +\frac{1}{6}[\frac{b^2}{a}\,\mathrm{arcosh}\,\frac{c}{b}+\frac{a^2}{b}\,\mathrm{arcosh}\,\frac{c}{a}] \\ = & \ 38.584m \end{aligned} \tag{31}$$

Finally, equation (11) gives the expected length a node travels with in this anchor zone:

$$\begin{aligned} \mathrm{E}[L] &= 2(56.6245) + \frac{1-0.4355}{0.4355}(38.584) \\ &= 163.262m \end{aligned} \tag{32}$$

For a node with a transmission range of d=10m, equation (15) gives R=0.00127N^2 in fluid limit conditions. Consequently, the criticality condition in (1) will be $1/\lambda$<123.95 s. It means in this hypothetical scenario, node arrival into the anchor zone on average every 123.95s is sufficient to float content without the need for additional infrastructure. Since fluid limit conditions do not hold in such small squares, equation (23) tells us the system should operate at $(1-p)/\lambda$ of the criticality threshold to achieve content penetration of p. For example, to achieve a 90% content penetration, (23) will compute to $1/\lambda$<12.4 s.

Hypothetical Squares	$P^{(\mathrm{exit})}$	$\mathrm{E}[L_e]$	$\mathrm{E}[L_{\mathrm{rwp}}]$	$\mathrm{E}[L]$	$1/\lambda$; p=90%
100 m × 100 m	0.436	76.52m	52.14m	220.62m	12.4s
250 m × 250 m	0.179	191.30m	130.35m	980.47m	39.2s
500 m × 500 m	0.101	382.60m	260.70m	3081.71m	97s

Table 1: Square Mobility metrics for different hypothetical squares

Similarly, assuming that all the squares introduced in the beginning of this section are bounded by a road and pedestrian side-walks of upto 13 m wide, floating content analysis and square mobility parameters for our squares can be computed. Table 1 summarizes these different quantities for our hypothetical squares, while table 2 gives the same for the three real-world squares, also introduced at the beginning of this section. We see from these metrics that as the size of the square gets

Real-World Squares	$P^{(\text{exit})}$	$\mathrm{E}[L_e]$	$\mathrm{E}[L_{\text{rwp}}]$	$\mathrm{E}[L]$	$1/\lambda$; p=90%
Piazza della Rep., Florence, Italy	0.579	56.62m	38.58m	141.30m	9.3s
Plaza Mayor, Madrid, Spain	0.422	85.96m	58.51m	252.05m	13.3s
Place Bellecour, Lyon, France	0.205	193.62m	131.71m	890.01m	34.2s

Table 2: Square Mobility metrics for three real-world squares

larger, *mean path length* within the anchor zone increases, thereby reducing the risk of sudden departure from the *anchor-zone* and relaxing more stricter conditions on smaller squares to support floating content applications.

5.4 Summary

In this chapter, we took the *floating content* concept as one example for urban content sharing schemes and analyzed its properties for city squares where fluid limit conditions are no longer applicable. First, we studied *Information availability* for small systems, where we set some primary objectives such as *content penetration* or *content availability* instead of simply having a content float somewhere in the anchor-zone without quantifiable availability. We found that to guarantee some reasonable information availability within the anchor zone, the system operating point should be higher than the threshold for larger systems. This new condition for achieving a targeted level of content penetration, p, is given by (23) whereas content availability is can be computed from (26).

We also studied the bootstrapping of the system and its characteristics especially during the initial transient stages. We found that for small systems, stochastic fluctuations can not be ignored as there is a clear danger of sudden content extinction, especially when the system is initialized for the first time. Therefore, successful *bootstrapping* of the system is essential which can be achieved, for example, by having the original *seed-node* pass its content to more users before departing.

6 The Opportunistic Network Environment Simulator

In this chapter, we will present the implementation of the Square Mobility model in the ONE simulator and study how it performs under in the floating content application. Before we go further in explaining how the square mobility model is implemented, we will first provide an overview of the Opportunistic Network Environment simulator, its features and basic operations. Next, we will study the internal software architecture of the ONE 1.4.1 which we used to implement the Square mobility model. We will give more emphasis to parts of the simulator that are relevant to extend its functionality and implement our own model. In the last part of this chapter, we will discuss the implementation of the Square mobility model in the ONE 1.4.1.

6.1 Overview

Studying DTN and opportunistic networks often involved running simulation tests. There are many different kinds of network and mobility simulators available out there such as *ns2* [45] and *OMNeT++* [46]. But these simulators are not suitable for DTN networks and protocols as they are not built with DTN characteristics in mind. Hence, results and analysis of DTN networks might not be always reliable. There are also others deemed more suitable to DTN such as dtnsim [47] and *dtnsim2* [5]. Yet again, these simulators are primarily built to simulate and test DTN routing algorithms. As they are only capable of simulating routing algorithms, usually a separate program (e.g. random processes that simulate contact times) or data set (e.g. real world traces from CRAWDAD [48]) is used to define certain mobility patters. Even in programs where reliable result can be found, usually quite many complicated steps are needed to set up the right simulation environment, collect relevant data and analyze afterwards. In chapter 3, we have discussed the limitation of such approaches as random processes might not always capture true movement patterns or that real world traces might be too small or unavailable with the desired mobility metrics (e.g. time granularity).

The Opportunistic Network Environment (ONE) [49] is a discrete event simulator (DES) built using the Java programming language specifically to address the lack of simulators for DTN and opportunistic networks. It also utilizes techniques of Agent-Based Simulation (ABS) for autonomous decision makings by a single or collection of agents . As such, it has combined different mobility models, DTN routing algorithms and easy and intuitive ways to collect and interpret simulation data into a single simulation environment. It is built in a modular fashion where different modules are responsible for similar set of actions such as the application module, interfaces module and movement module. Each module in ONE is used to emulate their respective environment so that a more dynamic and fine-tunable DTN simulation tests can be performed. In situation where it might be necessary

[5] http://shoshin. uwaterloo.ca/dtnsim2/

to use external data sources, the ONE offers features to import data or traces from sources such as ns-2. The outputs of individual modules, for example the movement module, can also be exported in suitable formats to be used by other simulators, in effect making the ONE a mobility simulator.

The *core package* of the program contains the main classes and interfaces of the simulator like the `DTNHost` (DTN capable host) and `ConnectionListener` (an interface for classes in the simulator that want to be informed about connection between hosts). The *input package* is the primary source that provides classes and interfaces for reading external data sources.

All message handling classes are grouped into the *Routing package*. This module has a wide variety of routing protocols such as the *Spray and Wait* [50], *Epidemic* [51] and *PRoPHET* [52]. The ONE also contains passive routers which are mainly used for interacting with other simulators, especially DTN routing simulators. Different movement models and their implementation are clustered into the *movement package* which is in charge of the movement behavior of nodes during simulation. It controls where a node should be at a given time or what its travelling speed is. For this purpose, it has different classes that implement a number of synthetic mobility models ranging from the simple ones like the *Random Walk* and *Random Way-Point* (RWP) mobility models to the more sophisticated Map Based Mobility (MBM) models like the *Shortest Path Map-Based Movement* (SPMBM) [53] that constrain movement of a node into some kind of predefined path or map. The *Working Day Movement* (WDM) [29] model that try to model human mobility pattern in a typical working day is also found in the this module. The mobility models in this module can either be used separately or in combination to emulate the intended user movement pattern. It is also possible to use external movement data such as real world traces or movement data generated by other simulators. In cases where existing mobility models are not sufficient, a separate module can be created that inherits the common behavior of other models and exhibit the desired movement pattern. This is in fact one of the great advantages the ONE offers.

During and after running any simulation test, it is very important to collect relevant statistics from the simulation and analyze the results. The reporting package contains all the report classes that generate valuable statistics such as message delivery probability or round trip times (RTT). The ONE has quite many reporting modules that either writes the figures into a suitable report file or some that actually digest and report the overall simulation. Similar to other packages in the ONE, it is also possible to add own reporting modules that fit the requirements of individual simulation scenarios

6.2 Running Simulations

Creating and running simulation scenarios in the ONE is typically made by writing a simple text based configuration file that defines a set of nodes, their movement model and the routing algorithm they are going to use. The configuration file should also contain other parameters such as transmission range and speed for the nodes, their buffer space and simulation time.

Once the appropriate scenario is created, the simulation can be run either in batch mode or in a GUI (Graphical User Interface) form. Running simulation in GUI can be helpful to visually analyze the simulation, while running in batch mode will only print out the advancement of the simulation in terminal. In both cases, report modules set in the configuration file will generate the relevant reports. In batch mode, the ONE also supports "Run indexing", a feature that allows running similar scenarios with a set of variables that should change in different runs. For example, the same simulation scenario can be run multiple times with different "*random generator seed value*" so that a more reliable conclusion can be drawn.

6.3 Software Architecture

Architecturally, the ONE is designed in a modular fashion where there are different packages with similar set of actions and they communicate using different interfaces of the simulator. Some of the main modules and packages of the ONE that are relevant to the implementation of the Square Mobility Model are:*Core Package*, *Input Package*, *Movement Package* and *Report Package*. In the following subsections, we will take a more deeper analysis of these main packages and study their future that help to extend the simulator for our needs. Figure (11) shows a simplified overview of the relationship between the main packages of the simulator.

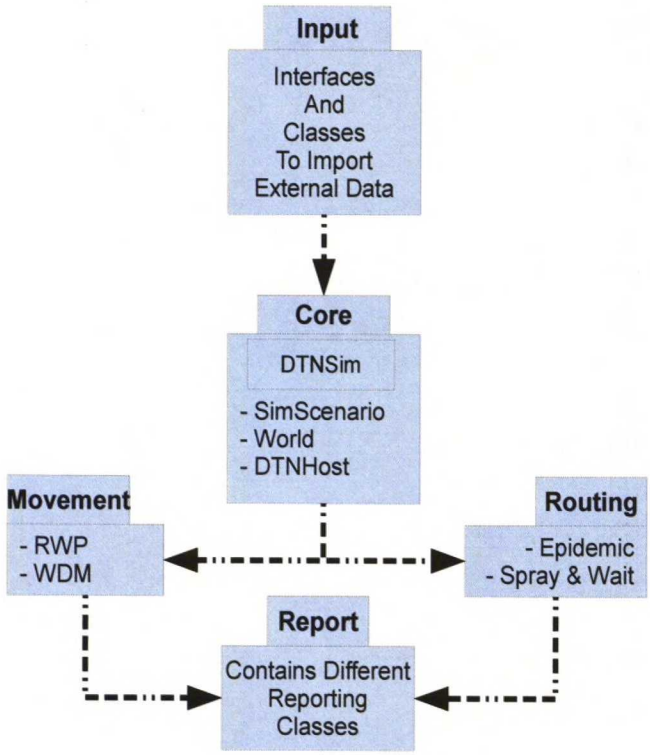


Figure 11: Simplified overview of the ONE Software Architecture

6.3.1 Core Package

As briefly mentioned in section 6.1, the ONE, written in Java, is an *agent* based discrete time event simulator where a time variable is incremented in fixed steps. The actual simulation world is abstracted in the `World` class which also hosts all *agents* of the simulator. The *agents* (wireless nodes) are `DTNHost` objects that have specific instance of movement model, wireless interface and other attributes. Thus, after each time increment, the class `DTNHost` will update coordinates of a node to determine its position and also check for any events, e.g. connection between nodes, and update the simulation world accordingly. These basic constituents of the simulator including the `DTNHost` are all packed into the *Core package*. For example, the simulator's main class, `DTNSim`, which starts up the SimScenario and World classes, is located in this package. The `SimScenario` class, also under this package, is responsible for storing and retrieving scenario settings. Additionally, the *core* offers interfaces for reporting about the occurrence of an event to other interested classes, either in the same or different packages.

6.3.2 Input Package

The ONE supports usage of external movement traces or other events in addition to those already implemented in the program. As such, interfaces and classes to import data from external sources are provided by the *Input Package*.

External event generating classes defined under the superclass `ExternalEvent` may define dynamic events which will be scheduled by the `ExternalEventsQueue` class and read by the `StandardEventsReader` class. Then the `EventsQueueHandler` will handle these events. The *Input package* has two main interfaces. The `External-EventsReader` interface, for example implemented by the `StandardEventsReader` class, provides interface for reading external events. The `EventQueue` interface is used by the simulator engine to query events and reference simulated *agents* (nodes) for appropriate action when the event is processed. Therefore, all event generators need to be created under this package and any class other than those in the Movement or Routing module that provide events for the simulator must implement the `EventQueue` interface.

6.3.3 Movement Package

The *movement package* is where different kinds of mobility models and their associated classes are implemented. The `MovementModel` is a superclass in this package that has the basic interface for getting the location of nodes or to request the next path if available. Then subclasses under the `MovementModel` utilize the interface to implement different mobility models. Based on certain mobility model, a series of way-points that the node follows constitute a path. Along with other packages, the movement package is dynamically loaded when the simulator starts. Therefore, all movement models need to be under this package and they all should extend the `MovementModel` class so they too can be dynamically added. The `Setting` class

under the core package can be used to stipulate which movement classes need to be loaded for a given scenario.

6.3.4 Report Package

The *report package* has quite many classes that can be used to collect and visualize simulation statistics and final outputs. As in the movement package, this package is also loaded dynamically into the simulator, for which the Report class is responsible. The `Report` class is a superclass (abstract) in the report package, and hence all reporting classes need to extend this superclass to be usable by the simulator. As a general rule, a reporting module needs to implement at least one interface to listen and record events. For example, a reporting module that collects data about connection between two nodes can implement the `ConnectionListener` interface from the *core package*.

6.4 Implementing the Square Mobility Model

The ONE 1.4.1, as it is, does not have the implementations of all the necessary features that create appropriate simulation for the Square mobility model. These features are:

- Open Simulation World: In order to simulate the openness of anchor zone, where a mobile node enters, spend some time and then leave, nodes should be added and removed from the simulation at run time.

- Arriving nodes can only be added into the simulation world at a set of pre-defined locations (*source*) and dynamically removed when they reach a set of pre-defined locations (exit region). A node cannot simply appear in a random way-point in the simulation world without first entering through one of the entry region/corner and travelling to that way-point by choosing it from a uniformly and randomly distributed way-points.

- Except a given set of nodes (usually a single seed node), other nodes should arrive into the simulation world (anchor zone) according to the Poisson arrival process with configurable rate. At the same time, the exit probability of nodes $P^{(\text{exit})}$ should be represented by a configurable variable in the simulation configuration file.

- The main performance metrics we want to collect from the simulation are the node's mean sojourn time, content floating probability or the latest time a given piece of content was relayed before it sinks, and the ratio of nodes that acquire the same copy of a content in proportion to the total number of nodes that visited the anchor zone when the simulation ends. Reporting classes that gather such information are not available in the current installation of the simulator.

Although the ONE 1.4.1 lacks the above mentioned features which are crucial features of the square mobility model, it at the same time offers an easy way to modify and implement own features into the simulator. Furthermore, the program for the ONE is released under an open source GPLv3 license, which makes it suitable to modify and added extra functionalities in the simulator. Therefore, to address the issues discussed in the beginning of this section, we have implemented the Square mobility model and related functionalities in version 1.4.1 of the ONE simulator under. Since different packages of the simulator are dynamically loaded when simulation starts [49], we can use these implementations in the configuration file and simulate the scenario "floating content in open city squares".

6.4.1 Open Simulation World

Normally, the simulation world in ONE is a closed region where nodes perform different movement patters based on some kind of mobility model. Although the ONE supports features to decide when nodes become active, i.e. perform movement and other DTN activities, or inactive where the node does not move and all interfaces and node capabilities are deactivated, it is not possible to either add new nodes while the simulation is running or remove existing ones. All nodes should be created at once when the simulator starts, and they stay until it ends. But, in open city squares, a node could enter or depart a region according to some arrival or departure function. Therefore, we need to extend the ONE so that we can dynamically add DTN nodes in run time, and also remove them when necessary.

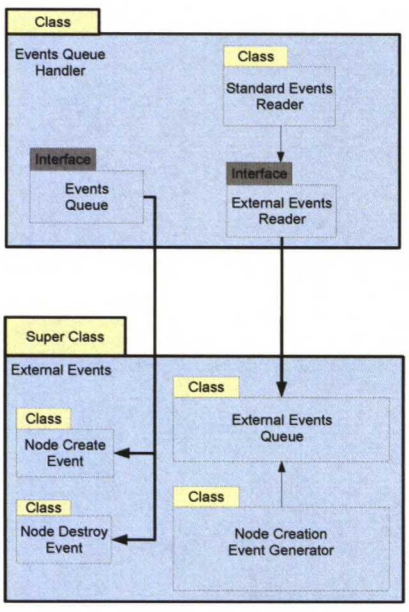


Figure 12: Input Package

The node creation and removal events during run time can be incorporated

into the core of simulator as external events. For dynamic node creation, we have implemented the class `NodeCreateEvent` (Appendix A) in the *Input package*, which extends the `ExternalEvent` superclass. This *event* will be called to create dynamic nodes into the simulation `world`. Additionally, we want the nodes to be created according to a Poisson process with configurable arrival rate. Therefore, we need an external event generator, the `NodeCreationEventGenerator` (Appendix B), which defines dynamic node creation events that are easily readable by the `StandardEventsReader` and integrated into the simulator. These events created by the `NodeCreationEventGenerator` can thus queue as any other external event does, and the `NodeCreateEvent` will be referenced by the `EventsQueue` interface when the time for node creation is there. Similarly, nodes can be removed from the simulation world by the `NodeDestroyEvent` (Appendix C), a class that also extends the `ExternalEvent` class. Events for node removal are scheduled by a method in the movement model which we will discuss next. Figure (??) depicts implementation of the Open Simulation World in the *input package*.

6.4.2 The Square Mobility Model

Basically, the **Square Mobility** model is a Random Way-point movement except that there is always a starting point for a node in the simulation world (entry corners of square shaped anchor zone), and nodes should be removed when they reach a certain boundary (*sink* of the anchor zone). In between entry and exit, a node simply roams from way-point to another making a zigzag path.

We implemented the Square Mobility model in ONE by a class under the Movement package called the `SquareMobility` (Appendix D). It extends the `MovementModel` class which is a super class for all movement models and it provides basic interface for acquiring node location and path. Since we want the nodes to start their journey within a set of predefined locations (entry points), the `SquareMobility` class has to override the `getInitialLocation()` Method. For this we built a new method, `initialCoord()`, with the modifier `Coord` which holds the 2D coordinates of node's location together with other arithmetic and alterations. This insures method returns a random initial placement for nodes with coordinates from the entry area rather than just a random placement from the whole simulation world. The entry area is a set of coordinates around the four corners of the square, from which the `getInitialLocation` returns node's initial placement. The entry area is set by the methods `entranceX()` and `entranceY()`. These methods define the entry area's width and length around the bottom left corner of the anchor zone.

To remove a node when it reaches a configurable exit zone, we also need to override the `getPath()` method, and call in a new method, `doexit()`,if the next way-point along the node's path leads to the exit. This is the method that schedules an exit event for the node and eventually gets the node removed. Simplified overview of how the `SquareMobility` works is shown in figure (13).

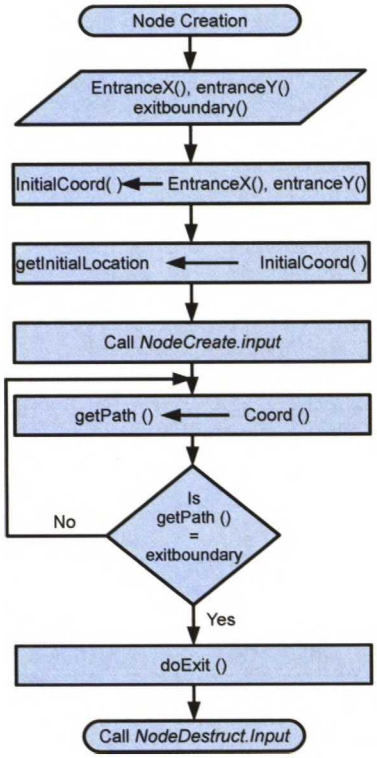


Figure 13: Flow Chart: Square Mobility

6.4.3 Reporting Classes

In the Reports package we have implemented two reporting classes; `FloatingDurationReport` (Appendix E) and `NodeSojournReport` (Appendix F) which are used to collect data about content floating probability and the mean sojourn time of a node respectively. The former works by keeping track of the latest time a given piece of content was relayed before it sinks. This reporting class is a replica of the original `MessageStatsReport` found in the original the ONE implementation with one extra variable (`timeRelayed`) added to hold the latest time a message was relayed and report the value. This variable is always updated whenever message transfer is successfully completed and the reporting class will write the final value as the latest time a message was relayed. Information about message transfers between hosts is provided by the `MessageListener`, an interface from the core package which is also implemented by the `FloatingDurationReport` reporting class.

The mean sojourn time of a node which is the time from its creation until it is removed from the simulation is reported by the `NodeSojournReport` class. This reporting class implements the `nodeListener` (Appendix G) from the core package which is an interface for classes that want to be informed about the status of a node. This interface has two methods (`nodeCreate` and `nodeDestruct`) that take

a `DTNHost` as their input parameter and called when a node is either created or destroyed. In `NodeSojournReport`, the specific time a node is removed from the simulation is recorded by the `nodeEnd ()` method which is called when a node destruction event is executed. This time is then used by the `getNodeLifeTime ()` method to calculate the node's sojourn time. In situations where the `nodeEnd ()` method is not called, for example when simulation time ends, then the `getNode-LifeTime ()` returns the difference of simulated seconds elapsed from the creation of a node until the simulation ends. In other cases, for example when the simulation is interrupted by a user, the current simulation time will be used.

6.5 Summary

In this chapter, we introduced ONE 1.4.1, the simulation environment we used to implement the *Square Mobility Model* and for our analysis on the "*floating content*" concept. We started by explaining the basic software architecture of the simulator and features that help extend its capability. Then we presented the implementation of the *Square Mobility Model* into the simulator. For the purpose of *floating content* analysis in city squares, we have also included two new reporting classes, `Floating-DurationReport` and `NodeSojournReport` for reporting *content floating duration* and the *mean sojourn time* of a node.

7 Simulation, Results and Discussion

In this chapter, we will present simulation results of floating content in open city squares and discuss our major findings. In order to verify the *square mobility* model reproduce the behaviour of nodes in open squares, we take the *mean sojourn time* as our metrice and simulate the movement of nodes in three different hypothetical squares and record node average sojourn time. The results will be compared with the theoretical value which also contribute in validating the proper implementation of the *square model* into the ONE simulation environment.

We analyze the conditions for increasing *content availability* in an anchor-zone by consideringr three different cases where the time the original seed node stays in the anchor-zone varies. We will give plots of content penetration for the three squares against node inter-arrival times. Properties of floating content during its initial transient stage and conditions of sucesfull *bootstrapping* for small systems will be studied by capturing content floating duration for different hypothetical and real-world squares of section 5.3. Before we go further, we will start by discussing the different simulation parameters and settings we use for the simulation.

7.1 Scenario Configuration

We analyze the square mobility model by considering the hypothetical and real-world squares discussed in section 5.3, floating content in open city squares, and validate against the result found in this section. We used the mean sojourn time of a node and duration of floating content as our metrics for validation.

The first analytical example in section 5.3 considers floating content in a square anchor zone of sides $100m \times 100m$, characterized by pedestrian exit probability $P^{(\text{exit})}$ of 0.436. In our simulation scenario configuration (Appendix H), we used a $100m \times 100m$ simulation world wrapped with an exit boundary 13m wide and a 2m by 5m entry area (*source*). Nodes will enter the simulation world through the *sink* at the bottom left corner of the square and move within the anchor zone by choosing their next way-point randomly. Way-points within the simulation world are equally likely to be chosen including those in the exit boundary. This will in effect give nodes a 0.4355 exit probability which is the same as our analytical example. The reporting class `NodeSojournReport` will record the mean sojourn time of a node starting from its entry until it gets removed.

All nodes in the simulation are equipped with a Bluetooth wireless interface with a transmission range of 10m and data transfer rate of 2Mbps. To simulate the floating content application, we explicitly allow only one node (seed node) to create a message of size 1M byte and share it among nearby nodes using the *epidemic* routing protocol. We will then keep track of this message as it gets transferred from node to node and record the latest time it was relayed using the `FloatingDurationReport` reporting class. Together with a certain target *lifetime*, *floating duration* can be used to calculate *floating probability*. The above simulation setup can be used for our hypothetical and real-world squares by simply adjusting the simulation world accordingly.

To study the effect of *bootstrapping* on content penetration, we will simulate the above scenario for three different cases where the seed node stays in the anchor zone for 600, 1500 and 3000 seconds before departing from the anchor-zone. Although the actual floating content scheme doesn't force the seed node to stay a certain amount of time before departing, doing so will help us investigate the case where a given piece of content might suddenly sink if a single carrier walks out from the anchor zone before passing its content to others. Finally, all of the above settings are repeated for wide range of pedestrian inter-arrival rate.

7.2 Results and Discussion

In this section, we will present results obtained by running simulations using scenario settings from section 7.1.

7.2.1 Mean Sojourn Time

We simulated and recorded mean sojourn time using `NodeSojournReport` for three hypothetical squares in section 5.3. Plot of the theoretical (blue) and simulated (red) mean sojourn time for a node according to the *Square mobility model* for these squares is shown in Figure (14). We can roughly say that the simulated mean sojourn time follows the theoretical one closely.The reason why the simulation result is a little higher can be accounted for two reasons. One is that, to make sure the seed node that publishes the first content stays the intended duration in the anchor zone, it is created as a *static* `DTNHost` node which can not be removed in run-time as the other *dynamic* nodes. In our implementation of the open city square mobility model, the ONE simulator allows destruction of `DTNHost` nodes that were dynamically added into the simulation and no other type of `DTNHost` nodes. Therefore, the seed node stayed for the whole simulation period of 10000s which is relatively too big to alter the *mean sojourn* time observed by `NodeSojournReport`.

The second reason is that, ideally *sources* and *sinks* are co-located (corners of the square) whereas in the simulator we cannot have the same area as a *source* and a *sink*. This is because it will lead us into the situation where all new nodes added into the simulation will be removed automatically since they will be in the exit region when created. For this, we implemented a small boundary around the simulation world as exit region. But we know from the property of *RWP* movement models, e.g. in [20,40], that random way-point nodes tend to concentrate around the center of a region and avoid the boundary. The same is true for nodes in the *Square Mobility Model*, see Figure 8 for example. This will have an effect of decreasing the probability of exit which in turn makes the average sojourn time higher than the theoretical value.

In fact, the formulæ in our analytical model are all closed-form solutions derived from generally accepted mathematical functions. Therefore, we can conclude that our analytical model captures the most important parameter, *mean sojourn time*, of the square mobility model.

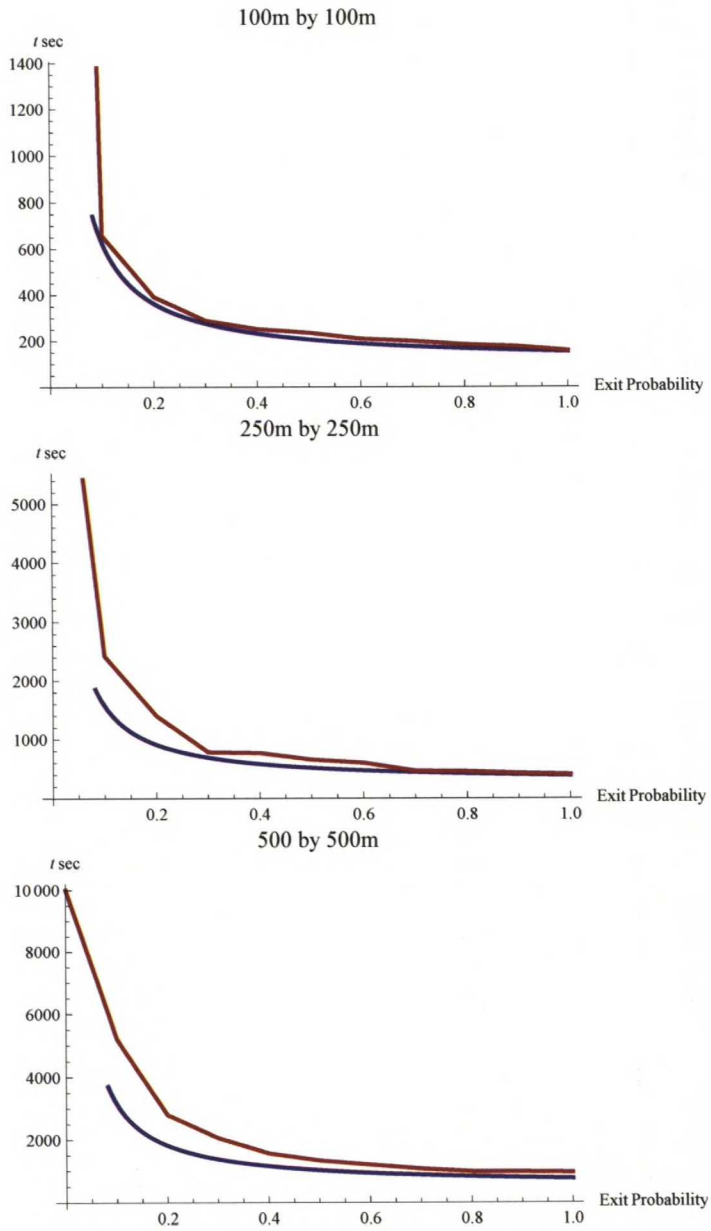


Figure 14: Plot of Theoretical (Blue) Vs Simulated (Red) Mean Sojourn Time for a A.) 100m by 100m B.) 250m by 250m C.) 500m by 500m

7.2.2 Content Availability

Figure (15) depicts the simulation results for *content penetration* against pedestrian inter-arrival rate for three different cases of seed node active times for the 100 m $\times$ 100 m square in section 5.3. By content penetration we mean the proportion of nodes that have the same copy of content with respect to total node population in the area. As discussed, the system is initialized with the seeder node staying in the anchor zone (active times) for $600s, 1500s$ and $3000s$ after which the content is left to float in the area by its own.

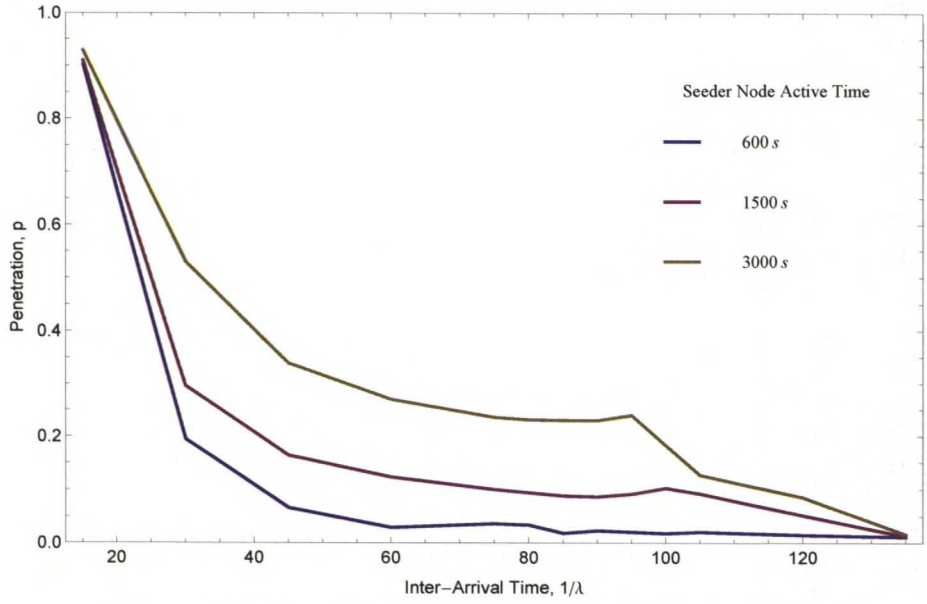


Figure 15: Simulation Result: Content Penetration, p

Figure (15) allows us to make two observations. First, when the system is initialized with the seeder node staying longer, better content penetration is achieved. This is especially true when the system is initiated for the first time, and only the seeder node has the content. This is because the longer the seeder node stays in the anchor zone, the more it passes content to other nodes consequently avoiding sudden extinction of content due to random exit behaviors. Therefore, to successfully *bootstrap* the system, the seeder node need to pass its content to more than one node, affirming our analysis of section 5.2. The other is, content penetration starts to increase when pedestrian inter-arrival times get shorter the criticality threshold for large systems. Around this criticality threshold, i.e. $1/\lambda$=123.23, penetration is minimal as content is simply residing somewhere in the anchor zone with very low probability of availability. On the other hand, above the criticality threshold, penetration starts to get higher thereby increasing content availability. For example, penetration of $P = 90\%$ is achieved for inter-arrival time of approximately $15sec$. This figure is in fact close to $12.4sec$ we computed in section 5.3. Therefore, operating the system beyond the criticality threshold clearly increase content availability, in-line with analytical findings.

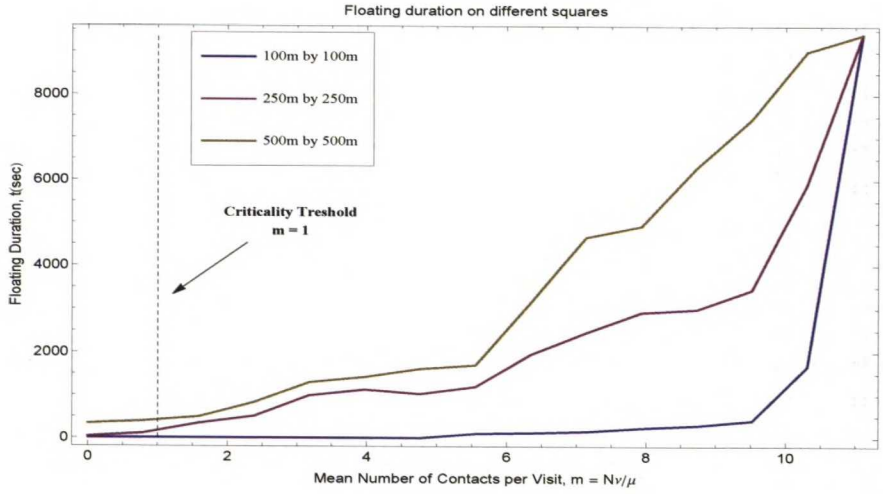
7.2.3 Content Floating Duration and Mean Remaining Lifetime

Floating duration as recorded by the `FloatingDurationReport` reporting class for the three hypothetical and three real-world squares from section 5.3 is shown in Figure (16.a) & (16.b) respectively.

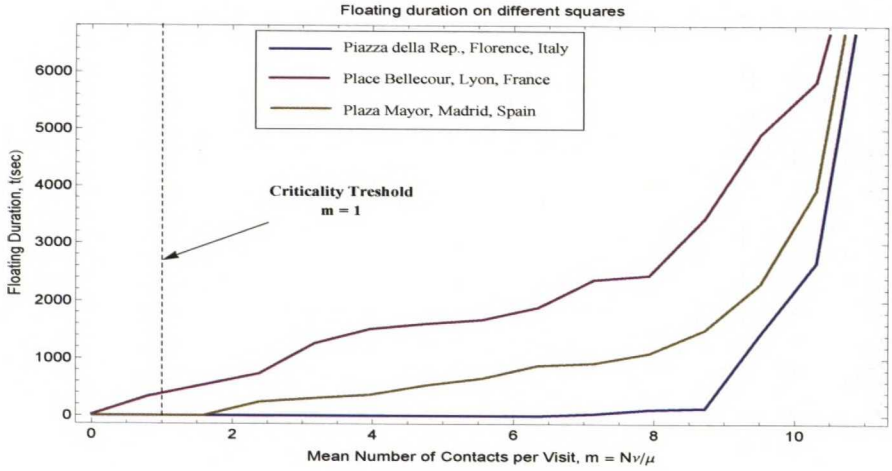
We can see from these plots that content floats when the mean number of contacts a random node experience per visit, $m = N\nu/\mu$, is larger than one [6]. But reasonably higher floating duration, for e.g. 1000s or 3000s, is achieved when the

[6]We say content has floated when `FloatingDurationReport` records $t > 0$.

system operating point is higher than the criticality threshold, $m = 1$. Additionally, we can see that smaller squares have lower floating duration as compared to larger squares indicating their sensitivity to stochastic fluctuations. But in all the squares, longer floating duration is achieved for $m > 10$.



(a) Floating Duration for a 100m by 100m, 250m by 250m and 500m by 500m hypothetical squares



(b) Floating Duration for real-world squares in Florence, Italy; Lyon, France and Madrid, Spain

Figure 16: Floating Duration for hypothetical and Real-world Squares of different sizes

The effect of operating the system beyond the criticality limit can also be visualized by studying the *Mean Remaining Lifetime* (MRL). In the context of *floating content*, we define Mean Remaining Lifetime (MRL), $m(t)$, as the expected extra lifetime a content will float given that it has already floated until time t. It can be used as a good indicator of the effect of bootstrapping floating content. MRL is a

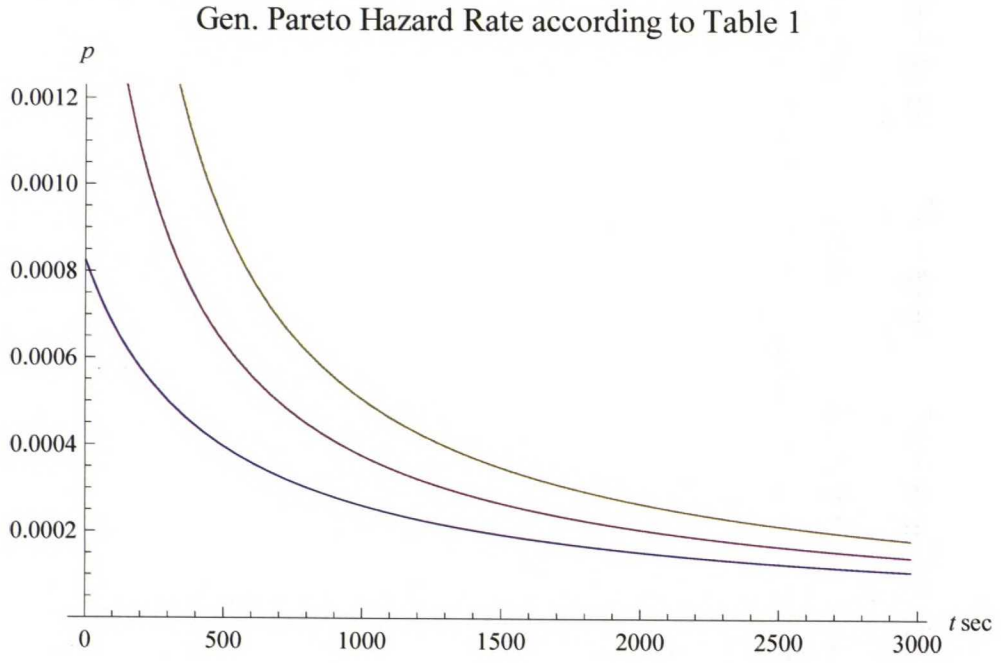


Figure 17: Gen. Pareto (Type II) Hazard Rate.

conditional probability given by:

$$m(t) = E(T - t|T > t) \tag{33}$$

Mean Remaining Lifetime is often expressed in terms of the Survival rate, $S(t) = 1 - F(t)$, as:

$$m(t) = \frac{1}{S(t)} \int_t^\infty S(t)\, \mathrm{d}t \tag{34}$$

$1/\lambda$ (sec)	K	α	μ
30	361.21	0.37874	-98.916
60	169.34	0.45722	-52.237
123.23	91.079	0.56449	-29.931

Table 3: Gen. Pareto (Type II) Values for selected values of inter-arrival times

We run a simple simulation test of floating content for the 100m by 100m square (section 5.3) where we observed floating duration for selected inter-arrival times for a maximum of $3000sec$. Fitting the observed Content lifetime distribution data using the *kolmogorov-smirnov* GOF test gives a *Generalized pareto distribution* (Type II) as best fit with parameters given in Table (3). In type II pareto distribution, μ is the location parameter, whereas K and α are the minimum value and shape parameters, respectively.

But we know that the survival function of a Pareto Type II distribution is given as:

$$S(t)=\begin{cases} (1+\frac{x-\mu}{k})^{-\alpha} & x>\mu, \\ 1, & \text{True} \end{cases} \tag{35}$$

In Figure (17), we have the Hazard Rate[7] for Generalized Pareto distribution of type II using the values from Table (1), and selected inter-arrival times of $123.23sec$, $60sec$ and $30sec$ (curves from top to bottom). In the figure, we see that the hazard rate generally decreases with time and level-off to a constant rate after a certain period of time. The top curve corresponds to a hazard rate for inter-arrival time of $123.23sec$ which is the critical inter-arrival time for large systems. We see that the hazard rate for the first few hundred seconds is very high and it takes longer time to level off, implying that content could easily sink due to stochastic fluctuations. But if we look at the case for inter-arrival time of $30s$, we see that the hazard rate is considerably lower, even from the very beginning at t=0.

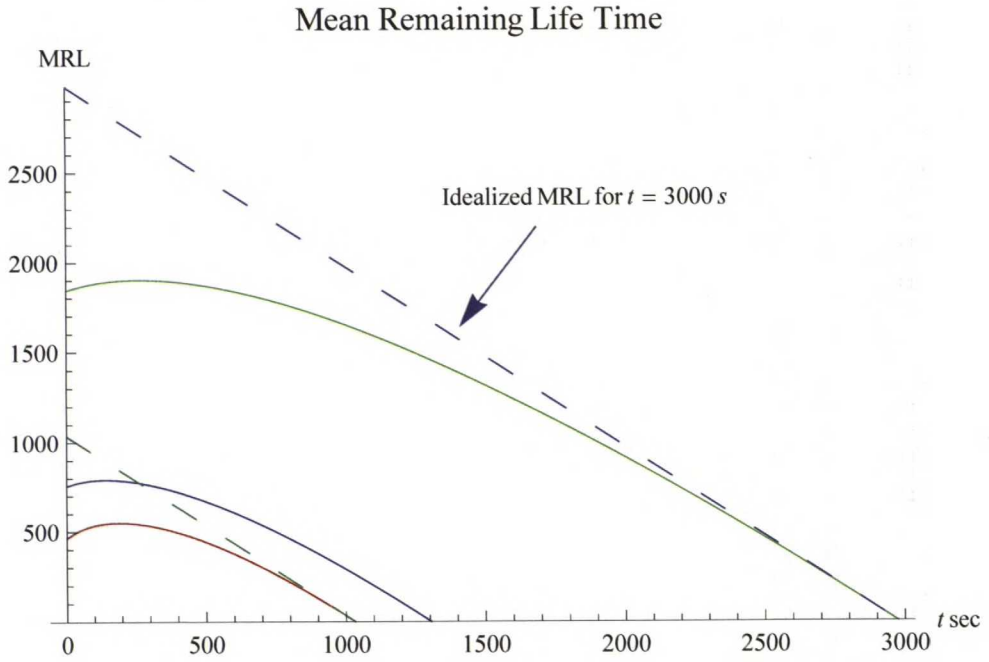


Figure 18: Mean Remaining Lifetime (MRL) for selected values of $1/\lambda = 30, 60, 123.23$ from top to bottom.

Finally figure (18) plots the corresponding MRL in comparison to an ideal system with deterministic lifetime of $1031s$ (the maximum achievable duration for $1/\lambda = 123.23$), which is a strictly decreasing linear MRL function. We see that during the initial transient (approximately the first $200s$) the mean remaining floating duration tends to increase before it starts to decrease linearly. It is also interesting to note that, although floating duration of $1031s$ is possible for inter-arrival rate of $123.23s$,

[7]The Hazard function (failure rate) is the ratio of the Probability Density Function (PDF) to the Survival function, S, which is used to describe the instantaneous failure rate.

the chances are it will sink before that. But if we look at the case for inter-arrival times of $60s$, we see that approximately after $t = 250s$, the curve lies above the line for the deterministic $1031s$ lifetime, but not until then. Hence, once the system is successfully bootstrapped to reach $t = 250s$, content is expected to float for the whole duration of $1031s$. Finally, the top curve corresponds to inter-arrival time of $30s$ which is well above the ideal MRL function of $1031s$ floating duration, and the system doesn't need bootstrapping.

Areas with higher arrival rates can almost immediately sustain floating content, whereas areas with lower arrival rates can support floating content if the system can avoid early extinction by operating beyond the criticality threshold. For areas with lower arrival rate, the question of how much above the criticality threshold should a system operate depends on the target lifetime of the system and on how long the original seeder node can stay in the anchor zone and successfully bootstrap the system or vise-versa.

7.3 Summary

In this chapter, we presented our simulation set up for running the *floating content application* using the *square mobility model* and validated our analytical findings of previous sections against simulation results. The main findings of simulation tests from this chapter indicate that small systems such as *city squares* have more stringent requirements for floating content and operating the system beyond the criticality threshold increases content availability, floating duration and avoids sudden and early content extinction, which is inline with out analytical prediction. We also showed that implementation of the *Square mobility model* into the ONE 1.4.1 produces similar node movement property, specifically sojourn time, as the theoretical one.

8 Conclusion

In this thesis work, we introduced and developed the *Square Mobility Model*, a *local pedestrian mobility model* for content sharing in urban areas . One prominent feature of the *square mobility model* is the *open mobility world* that allows pedestrians to freely enter the region, move around as they wish and finaly leave. Typical examples of such *open mobility world* are *city squares* and *parks* where content sharing schemes such as the floating content can be envisioned to be utilized. To better understand such scenario, we defined a square mobility model with one free parameter, $P^{(\text{exit})}$, that controls the shape of the path nodes take in the square before departing. Asymptotically, when $P^{(\text{exit})} \to 0$, one obtains the RWP model, and, when $P^{(\text{exit})} = 1$, the nodes make one stop and then leave the area. We have given several analytical results for the square mobility model that facilitate, e.g., adjusting the model parameters for simulation studies.

Additionally, our analysis of the *floating content* concept using the *Square mobility* model extends the understanding of *floating content* for small systems such as *city squares*. Previous efforts on floating content concentrated on studying the feasibility of floating content systems in the *fluid limit* (large systems). In particular, the criticality condition tells us under what circumstances the content avoids extinction, given it has been first distributed sufficiently well in the anchor zone. In contrast, in this thesis work, we studied:

1. Information availability (will a random node acquire it)
2. Initial *bootstrapping* properties, when the system is initiated for the first time (how soon will content sink after the original seed node departs). Here we note that the "thread of life" with an initially empty system (no other node than the seeder has the content) is indeed thin.

By analyzing different (stochastic) models capturing the essential characteristics of the system, we found that unlike very large systems, smaller systems can be very prone to stochastic fluctuations. This is because in a large system, the responsibility of storing and disseminating the content is effectively shared among a large number of nodes and thus it is sufficient that content replication occurs at a certain mean rate. That is, the nodes are collectively responsible for the operation. In contrast, when the system comprises a small number of nodes, averages are no longer sufficient but instead each node carrying the content is individually responsible for replicating it. Unfortunately, for small systems no obvious and comprehensive single criterion exists, but one has to choose some meaningful objective. One such first degree objective is to say, e.g., that the source node must be able to replicate the content at least to two other nodes.

8.1 Future Work

Possible future work includes more detailed analysis of both the initial content distribution phase and the square mobility model. The results of the *square mobility*

model can be extended into other general shapes with varying number and type of *sinks/sources*. It would be very initeresting to explore additional mobility parameters that can be used to implement a feedback system, for example about content penetration level. Although it is not possible to force nodes to stay in the anchor-zone beyond their will, such feedback systems can be used to warn original content holders about the possible danger of content extinction due to low penetration level. It would be also interesting to see if mobility parameters could be estimated automatically from a suitable video stream, thus enabling analysis of very long time periods.

References

[1] G. E. Moore, "Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff." *Solid-State Circuits Newsletter, IEEE*, vol. 11, no. 5, pp. 33 –35, sept. 2006.

[2] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '03. New York, NY, USA: ACM, 2003, pp. 27–34. [Online]. Available: http://doi.acm.org/10.1145/863955.863960

[3] F. P. Miller, A. F. Vandome, and J. McBrewster, *Delay-tolerant networking.* Alpha Press, 2010.

[4] "Defence advanced research project agency," http://www.darpa.mil/About/History/History.aspx, accessed: 31/12/2012.

[5] E. Hyytiä, J. Virtamo, P. Lassila, J. Kangasharju, and J. Ott, "When does content float? characterizing availability of anchored information in opportunistic content sharing," in *IEEE INFOCOM*, Shanghai, China, Apr. 2011, pp. 3123–3131.

[6] J. Ott, E. Hyytiä, P. Lassila, T. Vaegs, and J. Kangasharju, "Floating Content: Information Sharing in Urban Areas," in *Proc. of IEEE Percom 2011*, March 2011.

[7] J. Kangasharju, J. Ott, and O. Karkulahti, "Floating content: Information availability in urban environments," in *Proceedings of IEEE PerCom Workshops*, 2010, pp. 804–808.

[8] A. Villalba Castro, G. Di Marzo Serugendo, and D. Konstantas, "Hovering information: Self-organizing information that finds its own storage," School of Computer Science and Information Systems, Birkbeck College, London, UK, Tech. Rep. BBKCS707, Nov. 2007.

[9] O. Helgason, E. Yavuz, S. Kouyoumdjieva, L. Pajevic, and G. Karlsson, "A mobile peer-to-peer system for opportunistic content-centric networking," in *ACM/SIGCOMM Workshop on Networking, Systems, and Applications on Mobile Handhelds (MobiHeld)*, Aug. 2010.

[10] V. Lenders, M. May, G. Karlsson, and C. Wacha, "Wireless ad hoc podcasting," *ACM/SIGMOBILE Mobile Computing and Communications Review*, Jan. 2008.

[11] N. Thompson, R. Crepaldi, and R. Kravets, "Locus: A location-based data overlay for disruption-tolerant networks," in *Workshop on Challenged Networks*, Chicago, IL, Sep. 2010.

[12] E. Hyytiä, J. Virtamo, P. Lassila, J. Kangasharju, and J. Ott, "Characterizing availability of anchored information in opportunistic content sharing," Tech. Rep., Jul. 2010, http://www.netlab.hut.fi/~esa/tr-06-10.pdf.

[13] J "Floating content for probabilistic information sharing," *Pervasive and Mobile Computing*, vol. 7, no. 6, pp. 671 – 689, 2011, <ce:title>The Ninth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2011)</ce:title>. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574119211001210

[14] I. Rhee, M. Shin, S. Hong, K. Lee, S. Kim, and S. Chong, "On the levy-walk nature of human mobility," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 3, pp. 630–643, 2011.

[15] F. Bai, N. Sadagopan, and A. Helmy, "The important framework for analyzing the impact of mobility on performance of routing protocols for adhoc networks," *Ad Hoc Networks*, vol. 1, no. 4, pp. 383–403, 2003.

[16] Z. Haas, "A new routing protocol for the reconfigurable wireless networks," in *Universal Personal Communications Record, 1997. Conference Record., 1997 IEEE 6th International Conference on*, vol. 2. IEEE, 1997, pp. 562–566.

[17] F. Bai and A. Helmy, "A survey of mobility models," *Wireless Adhoc Networks. University of Southern California, USA*, vol. 206, 2004.

[18] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing*, vol. 2, no. 5, pp. 483–502, 2002. [Online]. Available: http://dx.doi.org/10.1002/wcm.72

[19] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. ACM, 1998, pp. 85–97.

[20] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 2. IEEE, 2003, pp. 1312–1321.

[21] E. Royer, P. Melliar-Smith, and L. Moser, "An analysis of the optimum node density for ad hoc mobile networks," in *Communications, 2001. ICC 2001. IEEE International Conference on*, vol. 3. IEEE, 2001, pp. 857–861.

[22] B. Liang and Z. Haas, "Predictive distance-based mobility management for pcs networks," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, mar 1999, pp. 1377 –1384 vol.3.

[23] C. Bettstetter, "Smooth is better than sharp: A random mobility model for simulation of wireless networks," 2001.

[24] ——, "Mobility modeling in wireless networks: Categorization, smooth movement, and border effects," *ACM Mobile Computing and Communications Review*, p. 2001.

[25] A. Jardosh, E. Belding-Royer, K. Almeroth, and S. Suri, "Towards realistic mobility models for mobile ad hoc networks," in *Proceedings of the 9th annual international conference on Mobile computing and networking.* ACM, 2003, pp. 217–229.

[26] S. Kumar, S. Sharma, and B. Suman, "Mobility metrics based classification & analysis of mobility model for tactical network."

[27] P. Jacquet, B. Mans, and G. Rodolakis, "Information propagation speed in mobile and delay tolerant networks," *CoRR*, vol. abs/0903.1157, 2009.

[28] W. Daamen, "Modelling passenger flows in public transport facilities," 2004.

[29] F. Ekman, A. Keränen, J. Karvo, and J. Ott, "Working day movement model," in *Proceeding of the 1st ACM SIGMOBILE workshop on Mobility models.* ACM, 2008, pp. 33–40.

[30] J. Crowther, Ed., *Oxford Advanced Learner's Dictionary*, 5th ed. Cornelsen & Oxford, 1998.

[31] J. Su, A. Chin, A. Popivanova, A. Goel, and E. D. Lara, "User mobility for opportunistic ad-hoc networking," in *In Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications*, 2004, pp. 41–50.

[32] D. Kotz and K. Essien, "Analysis of a campus-wide wireless network," *Wirel. Netw.*, vol. 11, no. 1-2, pp. 115–133, Jan. 2005. [Online]. Available: http://dx.doi.org/10.1007/s11276-004-4750-0

[33] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," *Comput. Netw.*, vol. 52, no. 14, pp. 2690–2712, Oct. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2008.05.003

[34] M. McNett and G. M. Voelker, "Access and mobility of wireless pda users," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 2, pp. 40–55, Apr. 2005. [Online]. Available: http://doi.acm.org/10.1145/1072989.1072995

[35] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket switched networks and human mobility in conference environments," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, ser. WDTN '05. New York, NY, USA: ACM, 2005, pp. 244–251. [Online]. Available: http://doi.acm.org/10.1145/1080139.1080142

[36] M. Kim, D. Kotz, and S. Kim, "Extracting a mobility model from real user traces," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, april 2006, pp. 1 –13.

[37] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa, "Stochastic properties of the random waypoint mobility model," *ACM/Kluwer Wireless Networks*, vol. 10, no. 5, Sep. 2004.

[38] W. Navidi and T. Camp, "Stationary distributions for the random waypoint mobility model," *IEEE Transactions on Mobile Computing*, vol. 3, no. 1, pp. 99–108, January-March 2004.

[39] J. L. Boudec, "On the stationary distribution of speed and location of random waypoint," *IEEE Transactions on Mobile Computing*, Dec. 2005.

[40] E. Hyytiä, P. Lassila, and J. Virtamo, "Spatial node distribution of the random waypoint mobility model with applications," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 680–694, Jun. 2006.

[41] E. Hyytiä and J. Virtamo, "Random waypoint mobility model in cellular networks," *Wireless Networks*, vol. 13, no. 2, Apr. 2007.

[42] E. Gilbert, *Random Plane Networks.* J.SIAM, December 1961, vol. 9, no.4.

[43] J. D. C. Little, "A proof of the queueing formula $L = \lambda W$," *Operations Research*, vol. 9, no. 3, pp. 383–387, 1961.

[44] P. A. P. Moran, *The statistical processes of evolutionary theory /Patrick A.P. Moran.* Clarendon Press, Oxford,, 1962.

[45] "The Network Simulator ns-2 (v2.1b8a)," http://www.isi.edu/nsnam/ns/, Oct. 2001.

[46] O. R. Helgason and K. V. Jónsson, "Opportunistic networking in omnet++," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, ser. Simutools '08. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 82:1–82:8. [Online]. Available: http://dl.acm.org/citation.cfm?id=1416222.1416315

[47] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '04. New York, NY, USA: ACM, 2004, pp. 145–158. [Online]. Available: http://doi.acm.org/10.1145/1015467.1015484

[48] "Homepage of the Community Resource for Archiving Wireless Data At Dartmouth (CRAWDAD)," http://crawdad.cs.dartmouth.edu/, 2007.

[49] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. New York, NY, USA: ICST, 2009.

[50] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, ser. WDTN '05. New York, NY, USA: ACM, 2005, pp. 252–259. [Online]. Available: http://doi.acm.org/10.1145/1080139.1080143

[51] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," Tech. Rep., 2000.

[52] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, Jul. 2003. [Online]. Available: http://doi.acm.org/10.1145/961268.961272

[53] A. Keränen and J. Ott, "Increasing reality for dtn protocol simulations," Helsinki University of Technology, Networking Laboratory, Tech. Rep., July 2007.

A Node Creation Event Generator

```
/*
 * Copyright 2013 Aalto University, ComNet
 * Released under GPLv3. See LICENSE.txt for details.
 */
package input;

import java.util.Random;

import cern.jet.random.AbstractDistribution;
import cern.jet.random.Poisson;
import cern.jet.random.Uniform;
import cern.jet.random.engine.MersenneTwister;
import cern.jet.random.engine.RandomEngine;

import core.DTNHost;
import core.Settings;
import core.SimScenario;

/**
 * Host creation -external events generator. Creates uniformly distributed
 * node creation events whose mobility model and inter-creation times can
 * be configured.
 */
public class NodeCreationEventGenerator implements EventQueue {


        /** Node creation time range -setting id ({@value}). Defines the
            time
         * range when messages are created. No nodes are created before the
             first
         * and after the second value. By default, nodes are created for the
         * whole simulation time. */
        public static final String NODE_TIME_S = "nodetime";
        /** Node creation interval range -setting id ({@value}). Can be
            either a
         * single value or a range (min, max) of uniformly distributed
         * random values. Defines the node inter-creation interval
             (seconds). */
        public static final String NODE_INTERVAL_S = "nodeinterval";
        /** Node ID prefix -setting id ({@value}). The value should be
            unique
         * for all node generators, so if you have more than one node
             generator
```

```
 * (e.g., with different mobility models or host groups), use
    different
 * prefix for all of them. The random number generator's seed is
    derived
 * from the prefix, so by changing the prefix, you'll get also a new
 * creation sequence. */
public static final String NODE_ID_PREFIX_S = "nodeprefix";

public static final String DISTRIBUTION = "distribution";
public static final String LAMBDA_S = "lambda";
RandomEngine engine = new MersenneTwister();
AbstractDistribution distribution;
String dist;

/** Time of the next event (simulated seconds) */
protected double nextEventsTime = 0;
/** */
static DTNHost prototypeHost;

/** Prefix for the nodes */
protected String idPrefix;
/** Interval between nodes (min, max) */
private int[] nodeInterval;
/** Time range for node creation (min, max) */
protected double[] nodeTime;

private double lambda;

/** Random number generator for this Class */
//protected Random rng;
protected int seed = 10;

/**
 * Constructor, initializes the interval between events,
 * node creation time, and node prefixes.
 * @param s Settings for this generator.
 */
public NodeCreationEventGenerator(Settings s){

        if (s.contains(NODE_INTERVAL_S)) {
                this.nodeInterval = s.getCsvInts(NODE_INTERVAL_S, 2);
        } else {
                this.nodeInterval = null;
                System.out.println("No Interval for Node Generation
                    " + s.contains(NODE_INTERVAL_S));
                System.out.println(s);

        }
```

```
if (this.nodeInterval.length == 1) {
        this.nodeInterval = new int[] {this.nodeInterval[0],
                        this.nodeInterval[0]};
}
else {
        s.assertValidRange(this.nodeInterval,
            NODE_INTERVAL_S);
}

if (s.contains(LAMBDA_S)) {
        this.lambda = s.getDouble(LAMBDA_S);
}

if (s.contains(NODE_ID_PREFIX_S)) {
        this.idPrefix = s.getSetting(NODE_ID_PREFIX_S);
        /* if prefix is unique, so will be the rng's
            sequence */
        this.seed = idPrefix.hashCode();
}
else {
        this.seed = 0;
}

if (s.contains(DISTRIBUTION)) {
        dist = s.getSetting(DISTRIBUTION);

        if(dist.equalsIgnoreCase("Poisson")){
                distribution = new Poisson(this.lambda,
                    engine);
                // System.out.println("DEBUG EventGenerator
                    Poisson distribution");
        }
        if(dist.equalsIgnoreCase("Uniform")){
                distribution= new
                    Uniform(this.nodeInterval[0],
                    this.nodeInterval[1], this.seed);
                // System.out.println("DEBUG EventGenerator
                    Uniform distribution");
        }
}



if (s.contains(NODE_TIME_S)) {
        this.nodeTime = s.getCsvDoubles(NODE_TIME_S, 2);
}
else {
        this.nodeTime = null;
```

```
        }

        /* calculate the first event's time */
        this.nextEventsTime = distribution.nextInt();
}

/**
 * Initiates the generator to create nodes with properties
 * (movementmodel, listeners, etc.) set according to a prototype
    model.
 * @param protoHost
 */
public static void init(DTNHost protoHost){
        prototypeHost = protoHost;

}


/**
 * Returns the next node creation event
 * @see input.EventQueue#nextEvent()
 */
public ExternalEvent nextEvent() {
        //return null;

        int interval = drawNextEventTimeDiff();

        // Create event and advance to next event
        DTNHost host = prototypeHost.getReplica();
        NodeCreateEvent nce = new
            NodeCreateEvent(this.nextEventsTime, host);


        this.nextEventsTime += interval;

        if (this.nodeTime != null && this.nextEventsTime >
            this.nodeTime[1]) {
                // next event would be later than the end time
                this.nextEventsTime = Double.MAX_VALUE;
        }

        return nce;
}


/**
 * Generates a (random) time difference between two events
 * @return the time difference
```

```
         */
        protected int drawNextEventTimeDiff() {
                //System.out.println("DEBUG: from distribution" +
                    distribution.nextInt() );
                return distribution.nextInt();

        }

        /**
         * Returns next node creation event's time
         * @see input.EventQueue#nextEventsTime()
         */
        public double nextEventsTime() {
                return this.nextEventsTime;
        }
}
```

B Node Create Event

```
/*
 * Copyright 2013 Aalto University, ComNet
 * Released under GPLv3. See LICENSE.txt for details.
 */
package input;

import core.DTNHost;
import core.Message;
import core.World;

/**
 * A message related external event
 */
public class NodeCreateEvent extends ExternalEvent {

	/* host that is created based on prototypehost*/
	private DTNHost host;

	/**
	 * Creates a Node to World with initial position
	 * @param
	 */
	public NodeCreateEvent(double nextEventsTime, DTNHost host){
		super(nextEventsTime);
		this.host = host;

	}

	public void processEvent(World world) {

		int size = 1;
		Message toGenerate = new Message(this.host, host,
		    ""+this.host+"-"+time, size);
		world.addHost(this.host);
	}


	@Override
	public String toString() {
		return "HOST created @" + this.time + " " + host.toString();
	}
}
```

C Node Destroy Event

```
/*
 * Copyright 2013 Aalto University, ComNet
 * Released under GPLv3. See LICENSE.txt for details.
 */
package input;

import core.DTNHost;
import core.World;

/**
 * External Event for node removal
 */
public class NodeDestroyEvent extends ExternalEvent {
	/** Time of the event (simulated seconds) */
	private DTNHost host;

	public NodeDestroyEvent(double time, DTNHost aHost) {
		super(time);
		this.host = aHost;
	}

	/**
	 * Processes the external event.
	 * @param world World where the actors of the event are
	 */
	public void processEvent(World world) {
		// this is just a dummy event
		world.removeHost(this.host);
	}

	/**
	 * Returns the time when this event should happen.
	 * @return Event's time
	 */
	public double getTime() {
		return this.time;
	}

	/**
	 * Compares two external events by their time.
	 * @return -1, zero, 1 if this event happens before, at the same
	    time,
	 * or after the other event
	 * @param other The other external event
```

```
	 */
	public int compareTo(NodeDestroyEvent other) {
		if (this.time == other.time) {
			return 0;
		}
		else if (this.time < other.time) {
			return -1;
		}
		else {
			return 1;
		}
	}

	/**
	 * Returns a String representation of the event
	 * @return a String representation of the event
	 */
	public String toString() {
		return "ExtEvent @ " + this.time;
	}

}
```

D Square Mobility Model

```
/*
 * Copyright 2013 Aalto University, ComNet
 * Released under GPLv3. See LICENSE.txt for details.
 */
package movement;

import input.NodeDestroyEvent;
import cern.jet.random.AbstractDistribution;
import cern.jet.random.Poisson;
import cern.jet.random.Uniform;
import cern.jet.random.engine.MersenneTwister;
import cern.jet.random.engine.RandomEngine;
import core.Coord;
import core.Settings;
import core.SimClock;
import core.SimScenario;

/**
 * The Open Square Mobility Model. Currently this is just a simple RWP
    movement,
 * with nodes entering the area (according to configurable inter-arrival
 * time defined in NodeCreationEventGenerator). The movement creates
 * zig-zag paths within the simulation area, and nodes exit the area when
 * entering a configurable exit-zone.
 */
public class SquareMobility extends MovementModel {

	/** how many waypoints should there be per path */
	public static final String DISTRIBUTION = "distribution";
	private static final int PATH_LENGTH = 1;
	private Coord lastWaypoint;

	RandomEngine engine = new MersenneTwister();
	AbstractDistribution distribution;

	/ * *
	 * a simple parameter to adjust the size of the exit area
	 */
	private double exitBoundary = 13;

	public SquareMobility(Settings settings) {
		super(settings);

		String dist;
```

```
        double lambda = 3.5;

        if (settings.contains(DISTRIBUTION)) {
                dist = settings.getSetting(DISTRIBUTION);
                if(dist.equalsIgnoreCase("Poisson")){
                        distribution = new Poisson(lambda, engine);
                        System.out.println("DEBUG Poisson
                            distribution");
                }
                if(dist.equalsIgnoreCase("Uniform")){
                        int min = 0; int max = 10; int seed = 10;
                        distribution= new Uniform(min, max, seed);
                        System.out.println("DEBUG Uniform
                            distribution");
                }
        }

}

protected SquareMobility(SquareMobility rwp) {
        super(rwp);
}

/**
 * Returns a possible placement for a host from an entrance area
     (source of the square)
 * @return Random position on the map
 */
@Override
public Coord getInitialLocation() {
        assert rng != null : "MovementModel not initialized!";
        Coord c = initialCoord();
        this.lastWaypoint = c;
        return c;
}

@Override
public Path getPath() {
        Path p;
        p = new Path(generateSpeed());
        p.addWaypoint(lastWaypoint.clone());

        Coord forDist = lastWaypoint.clone();
        Coord c = lastWaypoint;

        for (int i=0; i<PATH_LENGTH; i++) {
                c = randomCoord();
                p.addWaypoint(c);
```

```
        }

        /* if next coordinate will lead to exit, then schedule an
           exit event
         * for the node.
         */
        if(doExit(c)){

                double time = c.distance(forDist) / p.getSpeed();
                /* remove it*/
                SimScenario.getInstance().getWorld().scheduleEvent(
                                new
                                    NodeDestroyEvent(SimClock.getTime()+time,
                                    hostNode));
                }

        this.lastWaypoint = c;
        return p;
}

@Override
public SquareMobility replicate() {
        return new SquareMobility(this);
}

/*
 * use this to get a coordinate from entrance area, e.g., around a
    source.
 */
protected Coord initialCoord(){

        // Entry area/source around the bottom left corner of the
           anchor zone

        // about 1/20 th of the horizontal side length
        double entranceX = rng.nextDouble() * getMaxX()/20;

        // around vertical bottom, 1/40 times side length from bottom
        double entranceY = rng.nextDouble() * getMaxY()/40;

        // source will be 5m by 2m

        return new Coord(entranceX,entranceY);
}

protected Coord randomCoord() {
        return new Coord(rng.nextDouble() * getMaxX(),
                        rng.nextDouble() * getMaxY());
```

```
    }

    protected boolean doExit(Coord coord){
        // exit if close to boundaries of the area, i.e., walk our
           from all walls
        if(coord.getX()-exitBoundary<=0 ||
            coord.getX()+exitBoundary>=getMaxX() ||
                    coord.getY()-exitBoundary<=0 ||
                        coord.getY()+exitBoundary>=getMaxY()){
            return true;
        }else{
            return false;
        }
    }

}
```

E Floating Duration Report

```
/*
 * Copyright 2013 Aalto University, ComNet
 * Released under GPLv3. See LICENSE.txt for details.
 */
package report;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import core.DTNHost;
import core.Message;
import core.MessageListener;

/**
 * Report for generating different kind of total statistics about message
 * relaying performance including the latest time a message was relayed.
    Messages that were created during the warm up period
 * are ignored.
 * <P><strong>Note:</strong> if some statistics could not be created (e.g.
 * overhead ratio if no messages were delivered) "NaN" is reported for
 * double values and zero for integer median(s).
 */
public class FloatingDurationReport extends Report implements
    MessageListener {
        private Map<String, Double> creationTimes;
        private List<Double> latencies;
        private List<Integer> hopCounts;
        private List<Double> msgBufferTime;
        private List<Double> rtt; // round trip times

        private double timeRelayed;

        private int nrofDropped;
        private int nrofRemoved;
        private int nrofStarted;
        private int nrofAborted;
        private int nrofRelayed;
        private int nrofCreated;
        private int nrofResponseReqCreated;
        private int nrofResponseDelivered;
        private int nrofDelivered;
```

```
/**
 * Constructor.
 */
public MessageStatsReport2() {
	init();
}

@Override
protected void init() {
	super.init();
	this.creationTimes = new HashMap<String, Double>();
	this.latencies = new ArrayList<Double>();
	this.msgBufferTime = new ArrayList<Double>();
	this.hopCounts = new ArrayList<Integer>();
	this.rtt = new ArrayList<Double>();

	this.timeRelayed = 0;

	this.nrofDropped = 0;
	this.nrofRemoved = 0;
	this.nrofStarted = 0;
	this.nrofAborted = 0;
	this.nrofRelayed = 0;
	this.nrofCreated = 0;
	this.nrofResponseReqCreated = 0;
	this.nrofResponseDelivered = 0;
	this.nrofDelivered = 0;

}


public void messageDeleted(Message m, DTNHost where, boolean
    dropped) {
	if (isWarmupID(m.getId())) {
		return;
	}

	if (dropped) {
		this.nrofDropped++;
	}
	else {
		this.nrofRemoved++;
	}

	this.msgBufferTime.add(getSimTime() - m.getReceiveTime());
}
```

```
public void messageTransferAborted(Message m, DTNHost from, DTNHost
    to) {
        if (isWarmupID(m.getId())) {
                return;
        }

        this.nrofAborted++;
}


public void messageTransferred(Message m, DTNHost from, DTNHost to,
                boolean finalTarget) {
        if (isWarmupID(m.getId())) {
                return;
        }

        this.nrofRelayed++;
        this.timeRelayed = getSimTime();
        if (finalTarget) {
                this.latencies.add(getSimTime() -
                        this.creationTimes.get(m.getId()) );
                this.nrofDelivered++;
                this.hopCounts.add(m.getHops().size() - 1);

                if (m.isResponse()) {
                        this.rtt.add(getSimTime() -
                            m.getRequest().getCreationTime());
                        this.nrofResponseDelivered++;
                }
        }
}


public void newMessage(Message m) {
        if (isWarmup()) {
                addWarmupID(m.getId());
                return;
        }

        this.creationTimes.put(m.getId(), getSimTime());
        this.nrofCreated++;
        if (m.getResponseSize() > 0) {
                this.nrofResponseReqCreated++;
        }
}
```

```
public void messageTransferStarted(Message m, DTNHost from, DTNHost
    to) {
        if (isWarmupID(m.getId())) {
                return;
        }

        this.nrofStarted++;
}


@Override
public void done() {
        write("Message stats for scenario " + getScenarioName() +
                        "\nsim_time: " + format(getSimTime()));
        double deliveryProb = 0; // delivery probability
        double responseProb = 0; // request-response success
            probability
        double overHead = Double.NaN; // overhead ratio

        if (this.nrofCreated > 0) {
                deliveryProb = (1.0 * this.nrofDelivered) /
                    this.nrofCreated;
        }
        if (this.nrofDelivered > 0) {
                overHead = (1.0 * (this.nrofRelayed -
                    this.nrofDelivered)) /
                        this.nrofDelivered;
        }
        if (this.nrofResponseReqCreated > 0) {
                responseProb = (1.0* this.nrofResponseDelivered) /
                        this.nrofResponseReqCreated;
        }

        String statsText = "created: " + this.nrofCreated +
                "\nstarted: " + this.nrofStarted +
                "\nrelayed: " + this.nrofRelayed +
                "\nlatest_time_Relayed: " + this.timeRelayed +
                "\naborted: " + this.nrofAborted +
                "\ndropped: " + this.nrofDropped +
                "\nremoved: " + this.nrofRemoved +
                "\ndelivered: " + this.nrofDelivered +
                "\ndelivery_prob: " + format(deliveryProb) +
                "\nresponse_prob: " + format(responseProb) +
                "\noverhead_ratio: " + format(overHead) +
                "\nlatency_avg: " + getAverage(this.latencies) +
                "\nlatency_med: " + getMedian(this.latencies) +
                "\nhopcount_avg: " + getIntAverage(this.hopCounts) +
                "\nhopcount_med: " + getIntMedian(this.hopCounts) +
```

```
                    "\nbuffertime_avg: " +
                        getAverage(this.msgBufferTime) +
                    "\nbuffertime_med: " + getMedian(this.msgBufferTime)
                        +
                    "\nrtt_avg: " + getAverage(this.rtt) +
                    "\nrtt_med: " + getMedian(this.rtt)
                    ;

            write(statsText);
            super.done();
        }

}
```

F Node Sojourn Report

```
/*
 * Copyright 2013 Aalto University, ComNet
 * Released under GPLv3. See LICENSE.txt for details.
 */
package report;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Vector;

import core.DTNHost;
import core.Message;
import core.MessageListener;

import core.NodeListener;
import core.Settings;

/**
 * Reporting class for average node stay in simulation world
 * Report for generating different kind of total statistics about message
 * relaying performance and average node life time.
 * <P><strong>Note:</strong> if some statistics could not be created (e.g.
 * overhead ratio if no messages were delivered) "NaN" is reported for
 * double values and zero for integer median(s).
 */
public class NodesojournReport extends Report implements NodeListener {

	protected HashMap<DTNHost, NodeInfo> nodes;
	private Vector<Integer> nrofNodes;
	private List<Double> nodeAvgTime; // Average sjourne time


	/**
	 * Constructor.
	 */
	public NodesojournReport() {
		init();
	}

	@Override
	protected void init() {
		super.init();
```

```
        //this.creationTimes = new HashMap<String, Double>();
        this.nodes = new HashMap<DTNHost,NodeInfo>();
        this.nrofNodes = new Vector<Integer>();
        this.nodeAvgTime = new ArrayList<Double>();


}

/**
 * Objects of this class store time information about nodes.
 */
protected class NodeInfo {
        private double startTime;
        private double endTime;
        private DTNHost host;

        public NodeInfo (DTNHost aHost){
                this.startTime = getSimTime();
                this.endTime = -1;
                this.host = aHost;
        }

        /**
         * Should be called when the node exited to record the time.
         */
        public void nodeEnd() {
                this.endTime = getSimTime();
        }

        /**
         * Returns the time that passed between creation of this info
         * and call to {@link #nodeEnd()}. Unless nodeEnd() is
            called,
         * the difference between start time and current sim time
         * is returned.
         * @return The amount of simulated seconds passed between
            creation of
         * this info and calling connectionEnd()
         */
        public double getNodeLifeTime() {
                if (this.endTime == -1) {
                        return getSimTime() - this.startTime;
                }
                else {
                        return this.endTime - this.startTime;
                }
        }
```

```
        /**
         * Returns a string representation of the info object
         * @return a string representation of the info object
         */

    }


    @Override
    public void nodeCreate(DTNHost host) {
        // TODO Auto-generated method stub
        //System.out.println("Created: " + host.getAddress());
        nodes.put(host,new NodeInfo(host));
    }

    @Override
    public void nodeDestruct(DTNHost host) {
        // TODO Auto-generated method stub
        NodeInfo info = nodes.get(host);
        info.nodeEnd();
        //this.increaseTimeCount(info.getNodeLifeTime());
        //this.increaseResourceCount(host.getNrofMessages());
        //this.NodeInfo(info.getNodeLifeTime());
        this.nodeAvgTime.add(info.getNodeLifeTime());
    }



    @Override
    public void done() {
        write("Message stats for scenario " + getScenarioName() +
                      "\nsim_time: " + format(getSimTime()));

        String statsText = "\navg_sjourne: " +
            getAverage(this.nodeAvgTime)
                ;

        write(statsText);
        super.done();
    }

}
```

G Node Listener

```
/*
 * Copyright 2013 Aalto University, ComNet
 * Released under GPLv3. See LICENSE.txt for details.
 */
package core;

/**
 * Interface for classes that want to be informed about node movement.
 */
public interface NodeListener {

	/**
	 * Method is called every time a node is created.
	 *
	 * @param host that was created
	 */
	public void nodeCreate(DTNHost host);

	/**
	 * Method is called when a node is destructed.
	 *
	 *	* @param host that was destructed
	 */
	public void nodeDestruct(DTNHost host);

}
```

H Scenario Configuration file

```
#
# Default settings for the simulation
#
## Scenario settings
Scenario.name=FloatingContent_MovementModel.rngSeed%%lambda_%%Events1
Scenario.simulateConnections = true
Scenario.updateInterval = 1.0
# 43200s == 12h
Scenario.endTime = 10000
Scenario.nrofHostGroups = 3

# "Bluetooth" interface for all nodes
btInterface.type = SimpleBroadcastInterface
# Transmit speed of 2 Mbps = 250kBps
btInterface.transmitSpeed = 250k
btInterface.transmitRange = 10

# Common settings for all groups
Group.movementModel = RandomWaypoint
Group.router = EpidemicRouter
Group.bufferSize = 5M
# All nodes have the bluetooth interface
Group.nrofInterfaces = 1
Group.interface1 = btInterface
# Walking speeds
Group.speed = 1, 1
# Message TTL of 1000 seconds
Group.msgTtl = 10000
Group.nrofHosts = 1
Group.groupID = n

Group1.activeTimes = 0, 3000

Group2.dynamic = true
Group2.movementModel = SquareMobility
Group2.groupID = ped
Group2.nrofHosts = 1
Group2.speed = 1, 1

#To make sure the message is not delivered
Group3.activeTimes = 9998, 9999
```

```
## Movement model settings
# seed for movement models'
# pseudo random number generator (default = 0)
# 20 different seed numbers
# For run indexing, enter seed values separated by ";"
MovementModel.rngSeed = [1]
# World's size for Movement Models (width, height; meters)
MovementModel.worldSize = 100, 100

## Message creation parameters for the first group
# How many event generators
Events.nrof = 2

Events1.distribution = Poisson
# Class of the first event generator
Events1.class = NodeCreationEventGenerator
Events1.nodeinterval = 50,80

#Wide range of node inter_arrival rate
# For run indexing, enter inter-arrival values separated by ";"
Events1.lambda = [15]
Events2.class = ExternalEventsQueue
Events2.filePath = msgtrace.txt

## Reports - all report names have to be valid report classes
# how many reports to load
Report.nrofReports = 2
# Report classes to load
Report.report1 = MessageStatsReport2
Report.report2 = MessageStatsReport3
# Directory for storing the first report class
Report1.reportDir = [reports/../..]
# Directory for storing the second report class
Report2.reportDir = [reports/../..]

## Optimization settings -- these affect the speed of the simulation
## see World class for details.
Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder = true
```